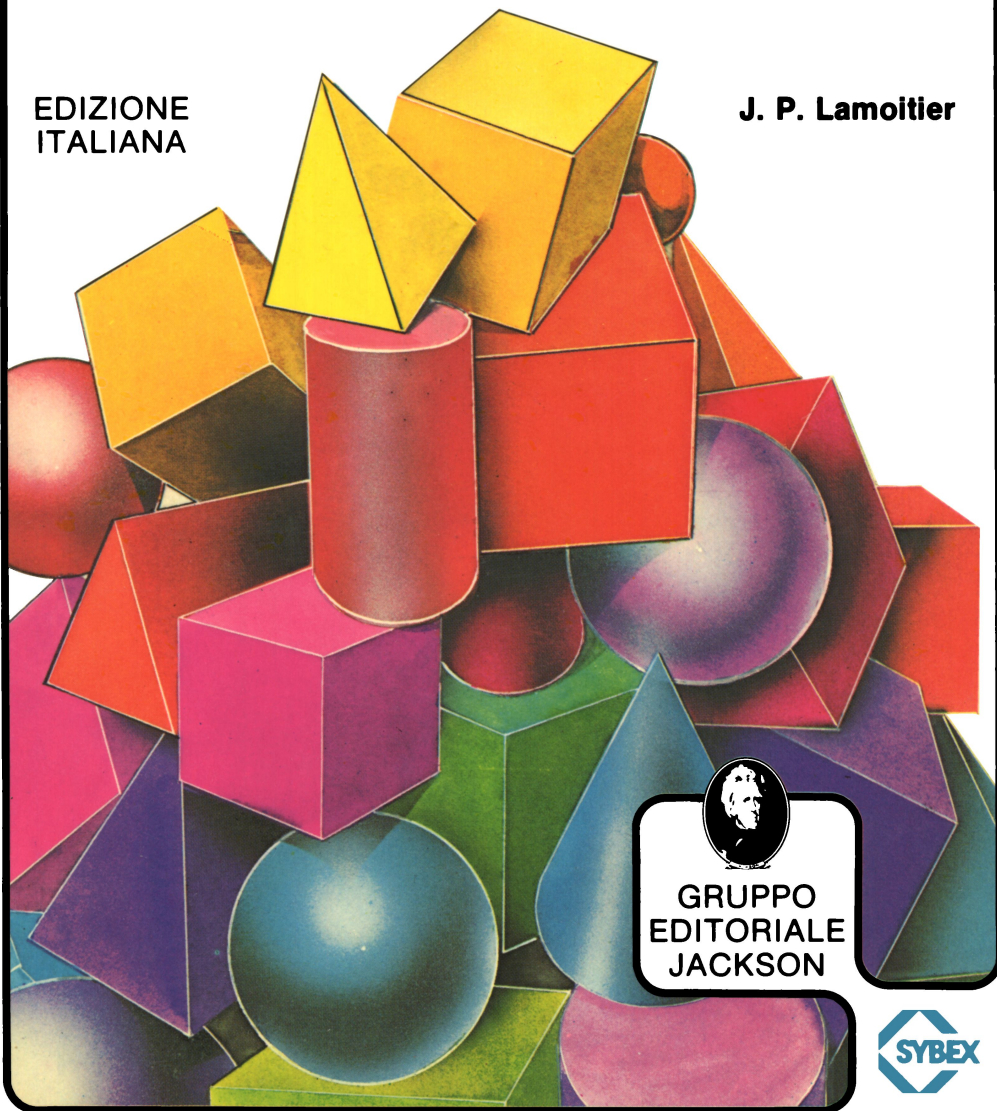
50 ESERCIZI IN
BASIC
EDIZIONE
ITALIANA
J. P. Lamoitier
GRUPPO
EDITORIALE
JACKSON
SYBEX

50 ESERCIZI IN

di
Jean-Pierre
Lamoitier

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

PREFAZIONE

Il linguaggio BASIC conosce attualmente un grande sviluppo data la diffusione di microcalcolatori personali e professionali.

Questo linguaggio è certamente il più diffuso in questo campo: è utilizzato per la gestione, l'acquisizione dei dati, i calcoli scientifici e per le applicazioni grafiche ed il controllo di processo.

Bene, la programmazione si impara essenzialmente con la pratica.

Ecco una collezione completa di esercizi concreti e completamente svolti che potrete seguire passo passo: enunciazione del problema, analisi, flow chart e commenti, programma, esempio di esecuzione.

Noi speriamo che dopo aver effettuato un numero sufficiente di questi esercizi e dopo averne letto la spiegazione, voi avrete effettivamente appreso "praticamente ... il BASIC".

Ciascun esercizio proposto è trattato in modo completo:

- enunciazione del problema
- analisi
- flow chart con commenti
- programma
- esempio di esecuzione

Questo metodo permette al lettore di verificare ad ogni passo quali progressi avrà compiuto nell'apprendimento dell'analisi di un problema. Mostra inoltre come scomporre il problema in "sotto-problemi" più semplici. Così è possibile risolvere separatamente ogni difficoltà e costruire un programma modulare di facile lettura e di facile adattabilità ad applicazioni concrete. Questi sotto-problemi possono in seguito servire da base per la costruzione di programmi più complessi.

Questo libro costituisce un complemento necessario ad una buona assimilazione del linguaggio. Un'appendice presenta in maniera coincisa le principali istruzioni del BASIC.

Alfine di rendere generalizzati i programmi l'autore ha utilizzato nella maggior parte dei casi, un BASIC molto diffuso (sotto-insieme dell'MBASIC della MICRO-SOFT). Per alcuni programmi sono proposte più versioni alfine di mostrare come sia possibile aggirare le lacune di certi BASIC.

SOMMARIO

CAPITOLO 0

IL VOSTRO PRIMO PROGRAMMA IN BASIC

0.0 INTRODUZIONE

Questo capitolo mostra al lettore che ciascuno può imparare a programmare partendo da esempi semplici e che, tranne per una certa categoria di problemi, la programmazione non é cosa da specialisti. Per comprendere questo capitolo, non é dunque necessario conoscere il BASIC e, partendo da un semplice esempio, il lettore apprenderà differenti categorie di istruzioni BASIC e, come sia possibile migliorare progressivamente un programma. Il tema scelto é il calcolo del bilancio familiare che seppur di limitato interesse é comunque utile per applicazioni più interessanti quali il calcolo dell'imposta sul reddito delle persone fisiche (IRPEF) che costituisce l'oggetto di due esercizi nel Capitolo 6.

Per migliorare la conoscenza del linguaggio, il lettore potrà da una parte leggere un libro teorico che presenti in maniera più precisa le differenti istruzioni del linguaggio, certamente in modo meno gradevole di questa, dall'altra fare degli esercizi che sono indispensabili nell'apprendimento della programmazione.

0.1 CALCOLO DEL BILANCIO FAMILIARE

Desideriamo ora calcolare il bilancio familiare di una famiglia partendo dal reddito imponibile e dal numero dei componenti della famiglia stessa. Chiameremo:

B bilancio familiare da calcolare
R reddito imponibile
N numero dei componenti la famiglia

Per ottenere il bilancio familiare B é sufficiente effettuare la divisione R/N. Proporremo quindi un programma in grado di svolgere le seguenti operazioni:

leggere R e N	`20 INPUT R,N`
calcolare $Q = \frac{R}{N}$	`30 Q=R/N`
	`40 PRINT Q`
stampare Q	`50 END`

Da questo piccolo programma potremo trarre le seguenti constatazioni:

— ogni linea comporta un numero di linea,
— ogni linea comporta un'istruzione,
— l'istruzione di lettura INPUT corrisponde all'impostazione di una informazione nel computer,
— l'istruzione di calcolo B=R/N si scrive su di una sola linea e la barra obliqua é il simbolo della divisione. Questa barra é chiamata "SLASH",
— il programma termina con l'istruzione END.

L'esecuzione del programma fornisce il seguente risultato:

```
? 100000,3
33333.3
```

Quando il calcolatore deve eseguire l'istruzione INPUT, stampa sul terminale un punto interrogativo che sta a significare l'attesa di dati da parte dell'operatore. In questo esempio di esecuzione, l'operatore ha battuto due numeri separati da una virgola: 100000 e 3. Poiché la lista delle variabili che segue INPUT é R,N, il primo valore é attribuito a R ed il secondo a N. Il computer effettua la divisione e la stampa del risultato.

Questo risultato é corretto dal punto di vista matematico, ma la presentazione é poco elegante. Comanderemo ora al computer di stampare un commento modificando leggermente il programma.

Per far stampare un testo, é sufficiente piazzarlo tra virgolette nell'istruzione PRINT. Cerchiamo ora di migliorare il programma stampando un commento all'inizio dello stesso.

```
10 PRINT "DARE I VALORI DI R ED N",
20 INPUT R,N
30 Q=R/N
40 PRINT "BILANCIO FAMILIARE = ";B
50 END
```

Questa virgola impedisce il passaggio alla linea seguente

Esempio di esecuzione:

```
DARE I VALORI DI DI R ED N  ? 100000,3
BILANCIO FAMILIARE = 33333.3
```

Quando il computer "attende" i dati, anticipa l'utilizzatore ponendo in generale un punto di domanda.

Con alcuni computer, in particolare con numerosi home-computer, si otterrà lo stesso risultato scrivendo:

```
20 INPUT"DARE I VALORI DI R ED N",R,N
```

In questa forma l'istruzione INPUT stampa un testo prima che il calcolatore legga i dati introdotti dall'operatore.

0.2 SECONDO CALCOLO DEL BILANCIO FAMILIARE

Consideriamo ora il caso dell'impiegato a stipendio fisso. I dati di base sono il reddito netto dal quale dedurremo forfettariamente un 10% per renderlo più reale, poi applicheremo le altre deduzioni quali: l'assicurazione sulla vita, interesse di un mutuo, etc., il calcolo del reddito imponibile é dunque:

reddito imponibile = 0.80 x reddito netto +pensioni e redditi vitalizi
a titolo gratuito - deduzioni

Il reddito imponibile é quindi utilizzato per il calcolo del bilancio familiare.

```
100 PRINT"IMPOSTARE IL REDDITO NETTO";
110 INPUT R1
120 PRINT"IMPOSTARE IL MONTANTE DELLE PENSIONI E LE RENDITE VITALIZIE";
130 INPUT P
140 PRINT"IMPOSTARE IL MONTANTE DELLE DEDUZIONI";
150 INPUT D
160 R=.80*R1+P-D
170 PRINT"IMPOSTARE IL NUMERO DEI COMPONENTI LA FAMIGLIA";
180 INPUT N
190 Q=R/N
200 PRINT
210 PRINT"REDDITO IMPONIBILE= ";R;" BILANCIO FAMILIARE= ";B
220 END
```

Esempio di esecuzione:

```
IMPOSTARE LA RENDITA NETTA ? 140000
IMPOSTARE IL MONTANTE DELLE PENSIONI E DELLE RENDITE VITALIZIE ? 1000
IMPOSTARE IL MONTANTE DELLE DEDUZIONI ? 7000
IMPOSTARE IL NUMERO DEI COMPONENTI LA FAMILIA ? 4
REDDITO IMPONIBILE= 109200
BILANCIO FAMILIARE=  27300
```

0.3 CONCLUSIONI

Si può ora modificare il programma per fare in modo che consideri:

- il numero di adulti ed anziani che incidono ciascuno per una certa parte
- il numero di bambini che incidono ciascuno per metà
- il reddito netto ordinando al computer di eseguire il calcolo delle deduzioni

Si può inoltre domandare al calcolatore di fare il calcolo del montante delle imposte. Questo sarà trattato nel Capitolo 6.

Con questo semplice esempio si potrà facilmente proseguire il lavoro. Per altri problemi invece, saranno necessarie nozioni più precise concernenti per esempio i flow-chart che saranno presentati in seguito.

Come tutti i linguaggi di programmazione, il BASIC é costituito da un "alfabeto" proposto nell'appendice.

CAPITOLO 1

FLOW-CHART

1.0 INTRODUZIONE

Il capitolo precedente ha mostrato come si possano facilmente apprendere i rudimenti del linguaggio BASIC, che ci ha permesso di scrivere un semplice programma. Tuttavia man mano che i problemi diventano più complessi, non é più possibile scrivere direttamente il programma. E' necessario prima analizzare il problema quindi disegnare un "flow-chart". In effetti l'esperienza ha mostrato che questo costituisce un aiuto prezioso per la programmazione, in particolare per chi é all'inizio di questa attività.

Lo scopo di questo capitolo é mostrare come si costruisce un flow-chart. I capitoli seguenti permetteranno al lettore di applicare ciò che avrà appreso ed inoltre di completare le proprie conoscenze.

In seguito, con l'esperienza, diviene possibile ridurre il tempo per disegnare il flow-chart, questo é comunque da sconsigliare al neofita.

1.1 SCOPO DEL FLOW-CHART

Il flow-chart costituisce una rappresentazione grafica del problema trattato. Lo stato attuale della tecnica non consente al calcolatore di comprendere questa rappresentazione, ciò pone, dunque, il problema della sua utilità.

Il flow-chart permette di verificare che al momento dell'analisi del problema, certi casi non siano stati dimenticati e facilita il dialogo tra diverse persone che partecipano alla elaborazione del programma. Per il neofita il flow-chart di dettaglio costituisce una tappa preliminare che aiuta ad una buona programmazione.

1.1.1 DIFFERENTI TIPI DI FLOW-CHART

Nella pratica si distinguono tre tipi di flow-chart:

— flow-chart di catena: soprattutto utilizzati in gestione, mettono in evidenza i legamenti tra archivi e programmi,

— flow-chart di principio: molto utili per descrivere in modo macroscopico grossi programmi che comportano un "concatenamento di algoritmi", é di interesse molto limitato per piccoli programmi,
— flow-chart di dettaglio: costituiscono una rappresentazione precisa e completa dell'argomento trattato, tolgono ogni eventuale ambiguità e permettono una facile programmazione.

Tuttavia il flow-chart deve restare il più possibile indipendente dal linguaggio di programmazione utilizzato.

1.1.2 NORME

Le tecniche del flow-chart sono state l'oggetto di una norma AFNOR Z 67-010 che é stata omologata nell'Aprile 1966. Alla fine del capitolo troveremo i principali simboli utilizzati per il disegno del flow-chart.

NOTE:

1. In italiano la traduzione del termine flow-chart é DIAGRAMMA DI FLUSSO.

2. Esistono altri metodi per la rappresentazione degli algoritmi, ma questi presuppongono un'approfondita conoscenza delle tecniche di programmazione.

Vediamo allora un semplice esempio di mini flow-chart, ciò permetterà di constatare che la soluzione non é una sola prima dello studio del problema.

1.2 IL VALORE PIU' ALTO DI DUE NUMERI A E B

Supponiamo che X assuma il valore più alto dei due numeri A e B. Come procedere per minimizzare la stesura?

1.2.1 Prima soluzione: si raffrontano A e B, se $A \geqslant B$ allora si pone il valore di A nella X altrimenti si pone il valore di B nella X.

Questo metodo può essere rappresentato per mezzo del flow-chart di Fig. 1.1 che contiene un "rombo" nel quale si trova il confronto $A \geqslant B$, e due "rettangoli" che corrispondono a delle "assegnazioni".

Il listato di Fig. 1.2 corrisponde alla sequenza BASIC relativa.

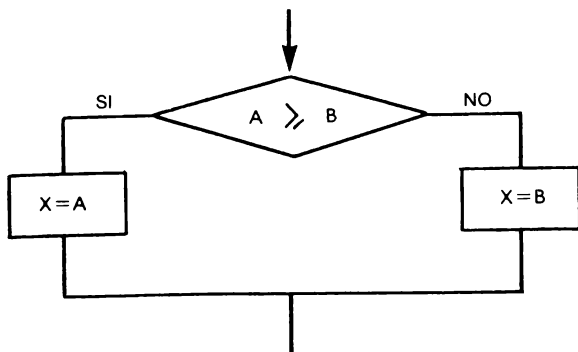



Figura 1.1 Figura 1.2

Utilizzando un BASIC più evoluto si potrà scrivere

```
100 IF A > = B  THEN X = A  ELSE X = B
```

1.2.2 Seconda soluzione: per evitare il salto della linea 120, si può modificare il flow-chart di Fig. 1.1 sostituendo una delle istruzioni di assegnazione, da qui il flow-chart di Fig.1.3 che potrà essere scritto in BASIC.

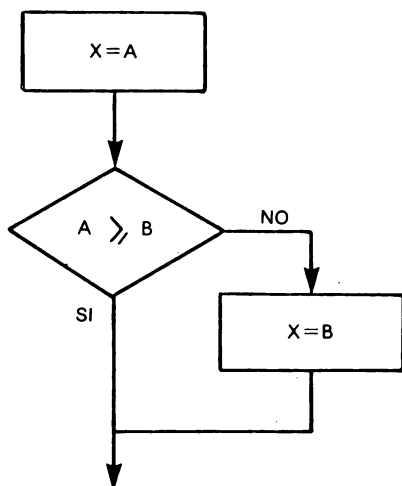



Figura 1.3 Figura 1.4

Sotto questa forma si economizza nel listato un GOTO. Questo permette di ridurne un pò l'ingombro.

Con un BASIC evoluto si avrà:

```
100 X = A
110 IF - B    A THEN X = B
```

1.2.3 Terza soluzione: certi interpreti BASIC, sfortunatamente rari, comportano le funzioni MAX e MIN. E' sufficiente allora scrivere:

```
100 X = MAX (A,B)
```

Attualmente queste funzioni MAX e MIN sono raramente disponibili sui personal computer.

NOTA: quando queste funzioni sono disponibili, ammettono un numero qualunque di parametri: si può quindi scrivere per esempio

```
Y = MAX (X,3,Z,C)
```

oppure ancora:

```
Y = MIN (X + Z,V*W,K*SIN(A))
```

1.3 ESEMPIO DI FLOW-CHART: ELEMENTO MAGGIORE DI UNA TABELLA

Supponiamo di cercare l'elemento più grande di una tabella A di 100 numeri. Il metodo proposto é il seguente:

- porre X = A(1)
- assegnare successivamente a I i valori 2,3,4 fino a 100:

    - confrontare X e A(I)
    - se X < A(I) trasferire il valore di A(I) nella X altrimenti continuare.

Quando si é terminato, X conterrà il valore più alto. Questo metodo può essere rappresentato con il flow-chart seguente

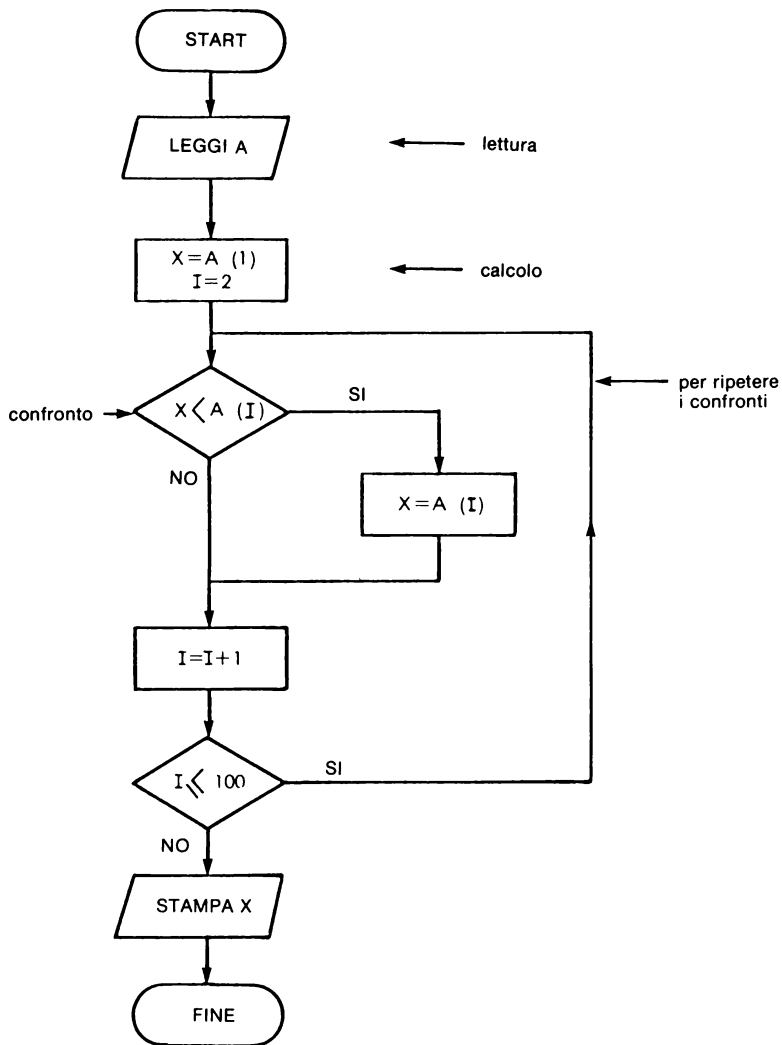



Figura 1.5

Questo schema pone in evidenza che:

— le istruzioni di Input/Output sono rappresentate da un parallelogramma,
— le istruzioni di calcolo sono rappresentate come un rettangolo
— le istruzioni di confronto sono rappresentate da un rombo.

Constatiamo inoltre un tipo di istruzione curiosa per l'"apprendista"

I = I + 1

In maniera più generale un'istruzione di calcolo si scriverà:

variabile = < espressione >

Questa istruzione significa calcolare il valore numerico dell'espressione e porre il risultato nella variabile situata a sinistra del segno = (uguale). Per questa ragione detta istruzione é chiamata di "assegnazione" ed il carattere = (uguale) rappresenta appunto il simbolo dell'assegnazione.

Per contro in un rombo I=100 significa confrontare I e 100 e vedere se essi hanno il medesimo valore. In nessun caso bisogna porre il valore 100 nella I. Nel rombo, il carattere = (uguale) é un simbolo di confronto.

1.4 COME VERIFICARE UN FLOW-CHART

Se si procede alla programmazione partendo da un errato flow-chart, il risultato non sarà certo quello auspicato. E' dunque consigliabile, dove possibile, assicurarsi dell'esattezza del flow-chart prima di cominciare la programmazione.

Per questo si può far "girare" il flow-chart "a mano", in questo caso l'essere umano simula la macchina che funzionerà nel rispetto di quanto previsto nel flow-chart potendo comunque intervenire ove fosse necessario.

Riprendiamo il flow-chart di Fig. 1.5 e interessiamoci ad una tabella più piccola, per esempio di 5 numeri.

I	1	2	3	4	5
A(I)	3	2	4	−1	6

Figura 1.6

All'inizio poniamo X = A(1), X assumerà quindi il valore 3. Possiamo ora far girare il loop di calcolo. La tabella 1.7 mostra l'evoluzione del contenuto di X in funzione di I.

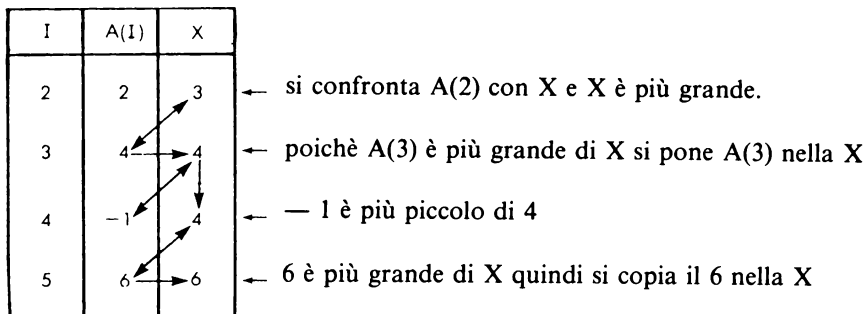



Figura 1.7

Constatiamo che X si sposta verso il valore più alto della tabella. Si può dunque programmare questo flow-chart.

NOTA: Questo metodo si può applicare solo con flow-chart molto semplici. Il flow-chart della Fig.1.5 può essere tradotto in BASIC in differenti modi, per esempio:

```
100 DIM A(100)
110 FOR I = 1 TO 100
120  READ A(I)
130  NEXT I
140 X = A(1)
50 FOR I = 2 TO 100
160   IF X > = A(I) THEN 180
170   X = A(I)
180   NEXT I
190 PRINT "MASSIMO NUMERO DELLA TABELLA= "; X
200 DATA ....
210 DATA ....
410 END
```

Figura 1.8

Questa forma non è la migliore, ma è di facile comprensione:

— le linee da 110 a 130 sono destinate a far leggere tutta la tabella,

— le linee da 140 a 180 inclusa, corrispondono alla ricerca del valore più alto,

— le linee da 200 a 400 contengono normalmente i valori numerici dei 100 elementi della tabella.

Critica a questo programma

Questo programma non può funzionare correttamente se la tabella non contiene esattamente 100 elementi. Di solito si preferisce far leggere prima il numero *n* degli elementi che costituiscono la tabella ed in seguito prevedere un programma capace di adattarsi al numero *n* di elementi. Il programma della Fig.1.9 è, da questo punto di vista, senz'altro migliore.

```
100 DIM A(100)
105 READ N
110 FOR I = 1 TO N
120   READ A(I)
130   NEXT I
140 X = A(1)
150 FOR I = 2 TO N
160   IF X > = A(I) THEN 180
170   X = A(I)
180   NEXT I
190 PRINT "MASSIMO NUMERO DELLA TABELLA= ", X
200 DATA 5
210 DATA 3,-2,34,5,0
410 END
```

Esempio di esecuzione:

```
MASSIMO NUMERO DELLA TABELLA= 34
```

Figura 1.9

Commenti:

— l'istruzione 105 permette di leggere il numero "*n*" degli elementi della tabella,

— la linea 200 contiene il valore numerico 5 corrispondente qui a 5 elementi,

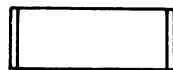
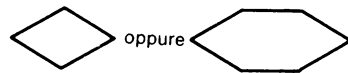
— la linea 210 contiene i 5 valori numerici,

— sotto questa forma, questo programma è limitato dall'istruzione DIM A(100) a 100 elementi. Modificando questa dichiarazione, è possibile adattare il programma a tabelle di maggiori o minori dimensioni,

— questa forma riduce le modifiche del programma consentendone inoltre una migliore leggibilità.

> In generale è consigliabile che il valore di chiusura del loop FOR, sia una variabile piuttosto che una costante.

Simbolo generale di trattamento

Chiamata ad un sottoprogramma

Simbolo di test — oppure

Rinvio

Inizio e fine programma — INIZIO — FINE

Input/Output
simbolo generale

Input da tastiera

Output su carta (stampante)

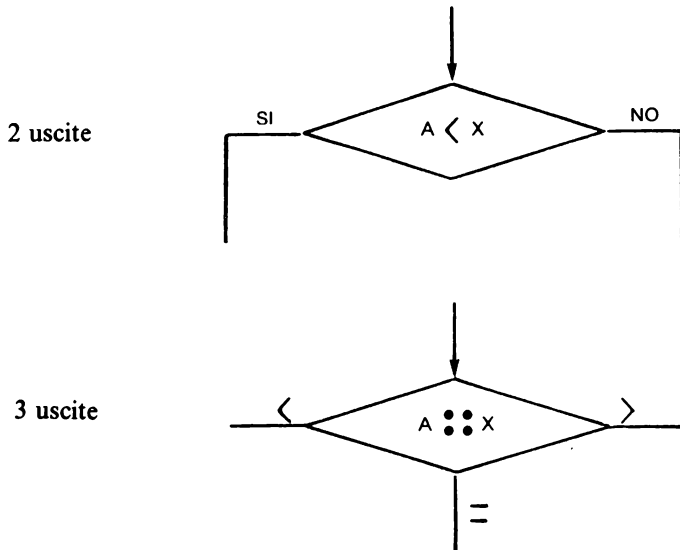
Un altro metodo più coinciso consisterà nello scrivere:

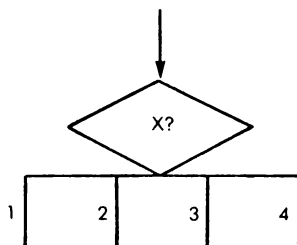
```
10 DIM A(100)
15 N = 100
20 MAT READ A
30 X = A(1)
40 FOR  I = 2 TO N
50 X = MAX(X,A(I))
60 NEXT I
70 PRINT":MASSIMO NUMERO DELLA TABELLA: ";X
80
90  )
.   |
.   > istruzioni DATA
.   |
160 )
170 END
```

attenzione la funzione MAX non è disponibile su tutti i sistemi (righe 30–60)

Figura 1.10

1.5 I TEST

In un flow-chart, un test ha un'entrata ed in generale 2 o 3 uscite. La rappresentazione si ottiene nel modo seguente:

Il simbolo :: è utilizzato come simbolo di confronto.

Si può concepire a livello di flow-chart un test con più di 3 uscite poichè il flow-chart è una rappresentazione dell'algoritmo. La norma non prevede la rappresentazione per test che abbiano più di 3 uscite. Si può per esempio utilizzare la seguente rappresentazione:

1.6 UNA TECNICA: LA BILANCIA

Come rappresentare un flow-chart di dettaglio in modo che la parte sinistra sia percorsa ad ogni loop dispari e che la parte destra sia percorsa ad ogni loop pari, fino al termine del loop stesso?

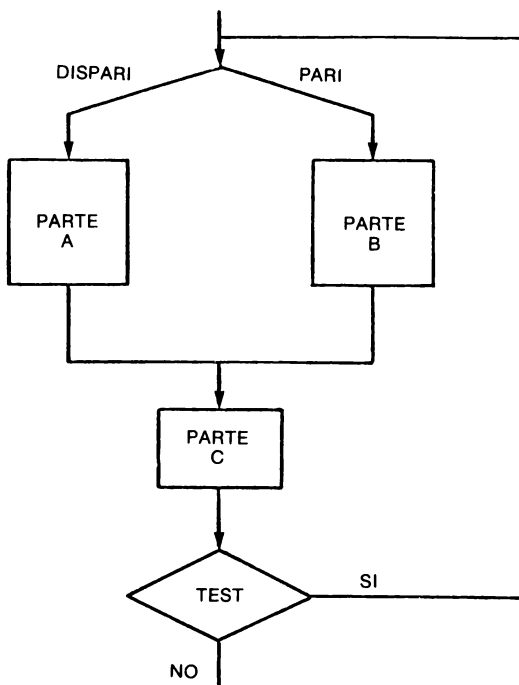



Figura 1.11

Un metodo semplice consiste nell'utilizzare una variabile ausiliaria, il cui valore permetterà di effettuare questa "bilancia". Per esempio si attribuirà alla variabile B il valore 0 prima di entrare nel loop. Successivamente un test sulla variabile B farà andare a sinistra se B vale 0. Nella posizione di sinistra si aggiungerà un'istruzione B=1 in modo che al test successivo ci si sposti nella parte destra. In questa parte si aggiungerà l'istruzione B=0 che per avere al giro seguente uno scambio verso sinistra. Questo porta al flow-chart di Fig. 1.12.

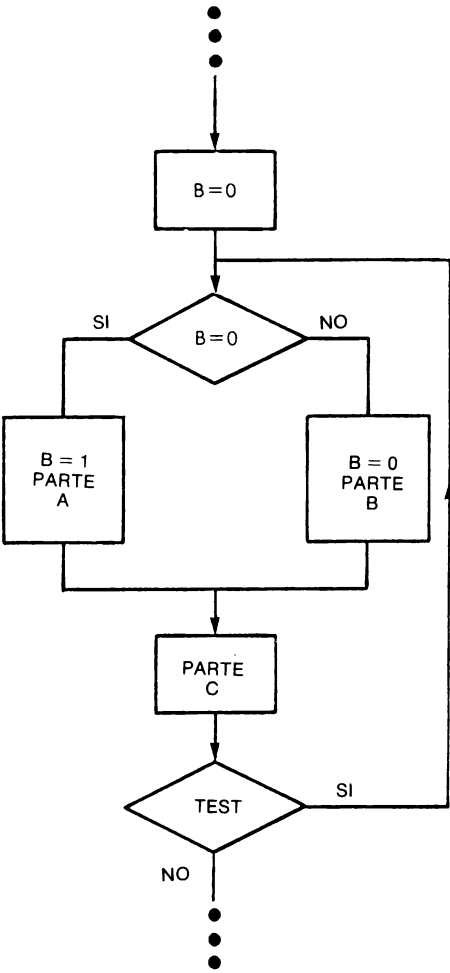



Figura 1.12

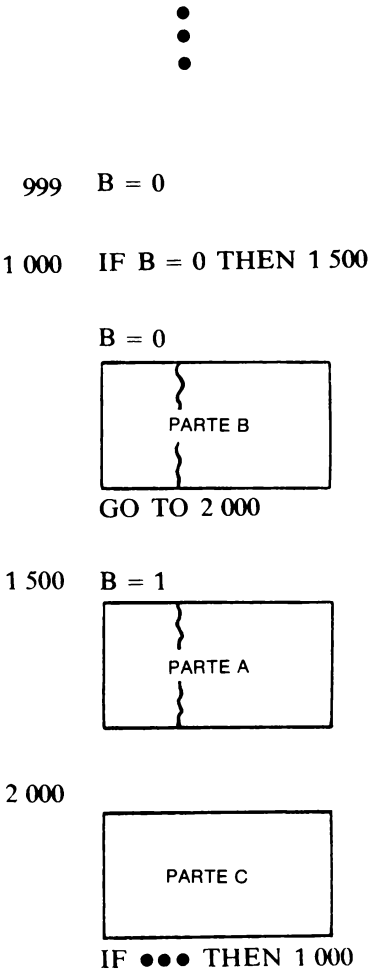



Figura 1.13

A partire dal flow-chart di Fig.1.7, la programmazione in BASIC è facilmente realizzabile nella sequenza mostrata dalla Fig.1.13. I numeri di linea sono dati solo a titolo di esempio.

NOTA 1: con il FORTRAN 77 si potrà scrivere:

```
  999 ||  B = 0
1 000 ||  IF (B.EQ.0)   THEN
      ||                      B = 1
      ||                      partie A
      ||                ELSE
      ||                      B = 0
      ||                      PARTIE B
      ||                ENDIF
      ||  Partie C
      ||  IF (      )   GO TO 1 000
```

Figura 1.14

NOTA 2: Con il CBASIC (1) si potrà scrivere:

```
  999     B = 0
1 000     IF B = 0      THEN
                              B = 1 :
                              Partie A
                        ELSE
                              B = 0
                              Partie B
          Partie C
          IF            THEN 1 000
```

Non abbiamo riportato i numeri di linea, che sono si obbligatori, ma, comunque non necessari alla comprensione.

Figura 1.15

1.7 CONCLUSIONI

In questo capitolo abbiamo visto gli elementi di base che permettono di tracciare un flow-chart. L'appendice presenta un caso generale la cui conoscenza non è utile per il "neofita" ma che è molto interessante per coloro i quali s'interessano di problemi più complessi. Studiando gli esercizi dei capitoli seguenti, il lettore potrà perfezionare la propria conoscenza dei flow-chart e soprattutto il modo per realizzarli.

(1) Il CBASIC costituisce il marchio depositato del BASIC esteso commercializzato dalla società americana SOFTWARE SYSTEMS e che funziona sotto il sistema operativo CP/M, disponibile sui sistemi compatibili INTEL 8080 Z80, INTEL 8080, INTEL 8085.

APPENDICE

Generalizzazione di una bilancia a p posizioni.

Bisogna ora che il primo percorso sia fatto nel ramo 1, il secondo nel ramo 2, il p-esimo nel ramo p, il $p+1$-esimo nel ramo 1 e così di seguito. In generale si dirà che il percorso dev'essere fatto nel ramo i modulo p.

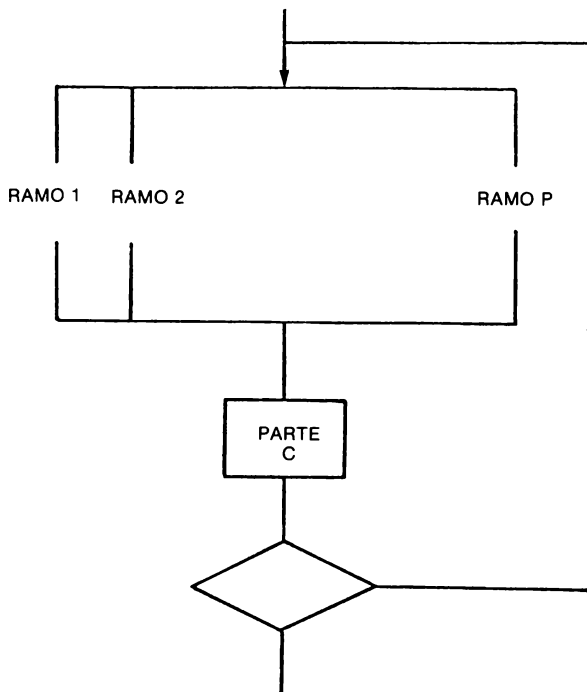



Figura 1.16

Non è più possibile rappresentare il metodo migliore in maniera coincisa per mezzo di un flow-chart, poichè sarà più opportuno utilizzare un'istruzione di

GOTO calcolato di Fig.1.17 anzichè una serie di test. Una prima soluzione è data dalla sequenza 1.17

```
  999 B = 1
1 000 ON  B GO TO 1 100, 1 300, 1 500, 1 600
1 100 B = 2
          Ramo 1
      GO  TO 1 800
1 300 B = 3
          Ramo 2
      GO  TO 1 800
1 500 B = 4
          Ramo 3
      GO  TO 1 800
1 600 B = 1
          Ramo 4
1 800
      Parte C
      IF ...THEN 1 000
```

Figura 1.17

Si constata allora che è possibile accorciare la sequenza utilizzando l'istruzione B = B + 1 posizionata nel "tronco" comune, tale istruzione fornisce la sequenza di Fig.1.18.

```
  999 B = 1
1 000 ON B GO TO 1 100, 1 300, 1 500, 1 600
1 100
          Ramo 1
      GO TO 1 800
1 300
          Ramo 2
      GO TO 1 800
1 500
          Ramo 3
      GO TO 1 800
1 660
          Ramo 4
1 800 B = B + 1
      IF B> 4 THEN B = 1
                Parte C
      IF ...THEN 1 000
```

Figura 1.18

CAPITOLO 2

ESERCIZI CONCERNENTI I NUMERI INTERI

2.0 INTRODUZIONE

Questo capitolo propone degli esercizi relativi ai numeri interi. I flow-chart non sono sempre facili da costruire e, da questo punto di vista, questi esercizi presentano un interesse didattico. Si consiglia al lettore, in caso di difficoltà, di non fermarsi troppo su certi esercizi, ma di passare agli altri capitoli per ritornarci in un secondo tempo.

Le soluzioni proposte valgono per gli interpreti BASIC "standard". Tuttavia sempre più frequentemente si trovano sistemi sui quali l'interprete BASIC accetta variabili intere, la cui rappresentazione si ottiene con una lettera seguita dal carattere %.

Esempio A%, B%

Per questi esercizi l'utilizzo di variabili intere che non superino il valore di + 32767 (1), permette di ridurre sia il tempo di esecuzione che lo spazio in memoria.

Comunque non tutti i sistemi accettano le variabili intere. Inoltre, insieme a queste variabili le funzioni standard SIN, COS, SQR, etc. sono raramente disponibili.

Una difficoltà che si incontra in questo tipo di problema è la necessità di effettuare delle "divisioni intere" e di calcolarne il resto. Per esempio si desidera conoscere Q e R tali che:

A = B x Q + R

Per questo bisogna effettuare la divisione intera A/B per la quale il BASIC non dispone di operatori particolari. Si utilizzerà dunque la funzione INT e si scriverà:

Q = INT (A / B)
R = A - Q x B

(1) Questo limite è proprio della maggior parte dei personal computer. Tale limite è più elevato quando si utilizzano dei "mini" o dei grossi calcolatori.

Così si otterà il divisore Q e il resto R. Quando si ha la possibilità di utilizzare delle variabili intere, è sufficiente scrivere:

```
Q % = A % / B %
R % = A % - B % * Q %
```

oppure se interessa solo il resto:

— nel primo caso:
```
R = A - B * INT ( A / B )
```
— nel secondo caso:
```
R% = A% - B% * ( A% / B% )
```

2.1 NUMERI INTERI COME $a^2 + b^2 = c^2$

Problema: Trovare tutti i numeri a e b interi compresi tra 1 e 100 in modo che $a^2 + b^2$ sia uguale ad un quadrato perfetto.

Si chiede:

— di analizzare il problema,
— di stabilire un metodo e disegnare il flow-chart,
— di scrivere il programma BASIC corrispondente.

Soluzioni: Per prima cosa notiamo che sarà necessario eliminare una delle due soluzioni identiche salvo una permutazione dei valori. Per esempio:

$$\begin{cases} a = 3 \\ b = 4 \\ c = 5 \end{cases} \quad e \quad \begin{cases} a = 4 \\ b = 3 \\ c = 5 \end{cases} \qquad \text{costituiscono due soluzioni identiche}$$

per evitare questo si cercheranno dei numeri b che siano $> a$. Si potrà quindi far mutare una variabile intera i da 1 a 99 e la seconda variabile j da $i + 1$ a 100.

Bisogna ora sapere se $i^2 + j^2$ è un quadrato perfetto, vengono presentati due metodi:

1° metodo. Si procederà con prove successive su una variabile k alla quale si faranno assumere dei valori numerici partendo da $j + 1$.

Se si trova $i^2 + j^2 = k^2$ si è ottenuta una soluzione

Se si trova $i^2 + j^2 > k^2$ proseguire le prove incrementando k di 1

Se si trova $i^2 + j^2 < k^2$ la coppia i, j è errata.

Flow-chart

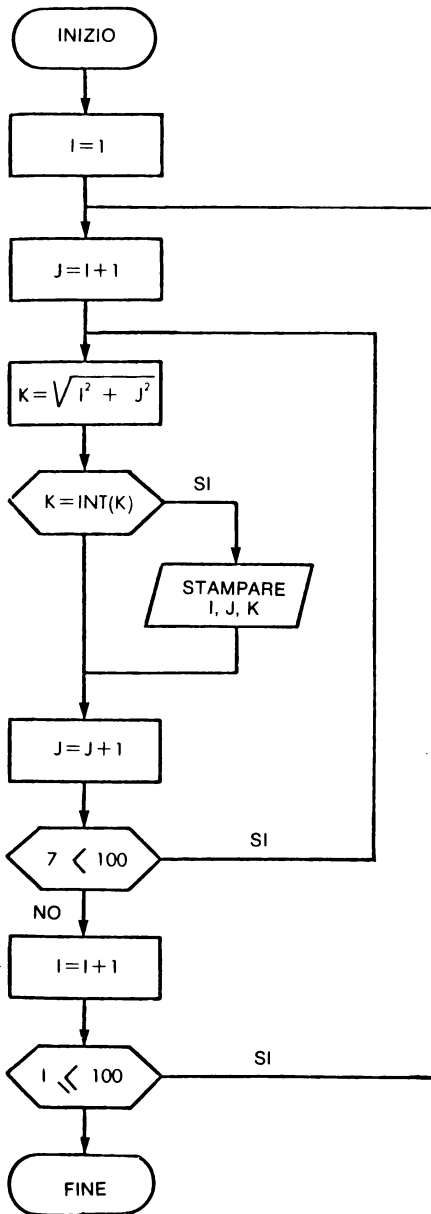
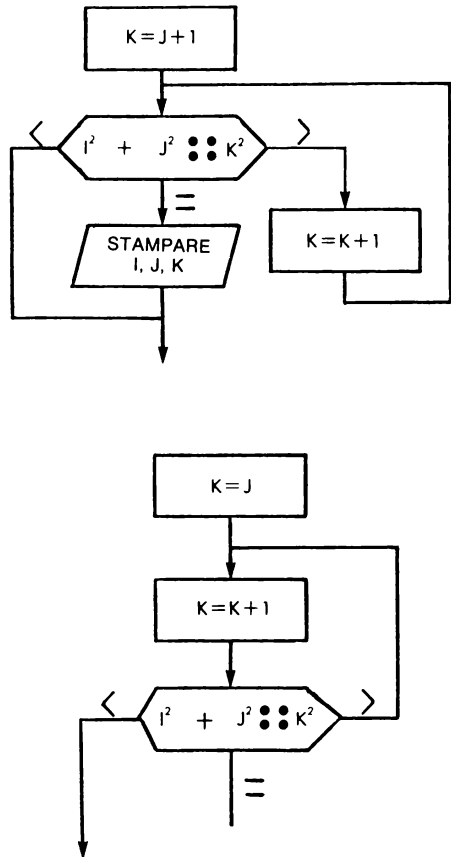



Figura 2.1

Figura 2.2

2° metodo. Si calcola:

$$k = \sqrt{i^2 + j^2}$$

se k è intero allora si è trovata una soluzione, altrimenti la coppia i, j è errata. Per sapere se k è intero, è sufficiente comparare k ed INT (k). Siamo così giunti ai flow-chart Figg.2.1 e 2.2 nei quali constatiamo:

— la presenza di un loop di calcolo su J per fare variare J da I+1 a 100,

— la presenza di un loop di livello superiore per fare variare I da 1 a 99.

Partendo dai flow-chart di Figg.2.1 e 2.2, si potranno facilmente scrivere i programmi riportati in Figg.2.3 e 2.4.

lista del programma con il secondo metodo

```
 5 N=100
10 FOR I = 1 TO N-1
20     J=I+1 TO N
30     K=SQR(I*I+J*J)
40     IF K <> INT(K) THEN 60
50        PRINT I;J;K
60     NEXT J
70 NEXT I
80 END
```

Figura 2.3

lista del programma con il primo metodo

```
  5 N=100
 10 FOR I = 1 TO N
 20   FOR J = I+1 TO N
 30       S=I*I+J*J : K=J
 40       K=K+1 : K2=K*K
 50       IF K2 < S THEN 40
 60       IF K2 > S THEN 80
 70          PRINT I;J;K
 80       NEXT J
 90   NEXT I
100 END
```

Figura 2.4

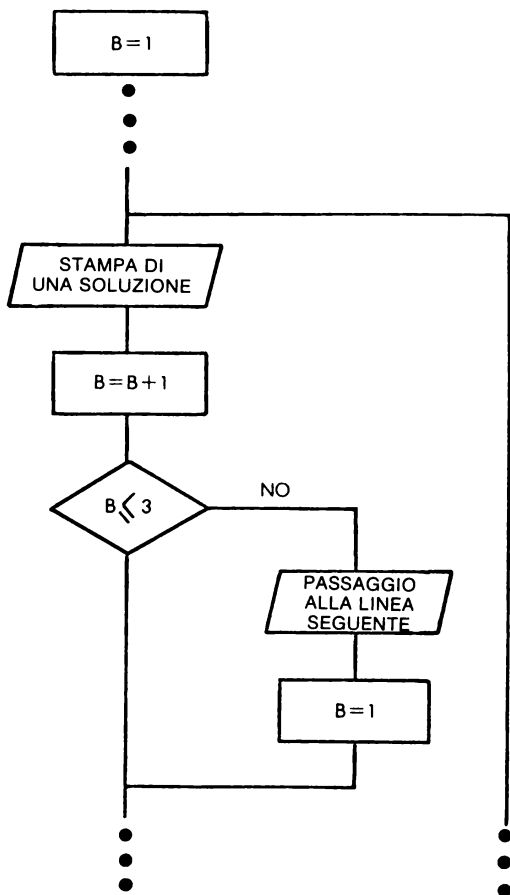
Per facilitare la lettura di un programma, consigliamo al lettore di scrivere in "modo strutturato" scalando di circa due caratteri le istruzioni riguardanti i loop FOR ... NEXT.

I listing delle Figg.2.5 e 2.6 mostrano l'utilità dell'istruzione PRINT USING che permette di migliorare la presentazione dei risultati.

```
  3   4   5
  5  12  13
  6   8  10
  7  24  25
  8  15  17
  9  12  15
  9  40  41
 10  24  26
 11  60  61
 12  16  20
 12  35  37
 13  84  85
 14  48  50
 15  20  25
 15  36  39
 16  30  34
 16  63  65
 18  24  30
 18  80  82
 20  21  29
 20  48  52
 20  99 101
 21  28  35
 21  72  75
 24  32  40
 24  45  51
 24  70  74
 25  60  65
 27  36  45
 28  45  53
 28  96 100
 30  40  50
 30  72  78
 32  60  68
 33  44  55
 33  56  65
 35  84  91
 36  48  60
 36  77  85
 39  52  65
 39  80  89
 40  42  58
 40  75  85
 40  96 104
 42  56  70
 45  60  75
 48  55  73
 48  64  80
 48  90 102
 51  68  85
 54  72  90
 56  90 106
 57  76  95
 60  63  87
 60  80 100
 60  91 109
 63  84 105
 65  72  97
 66  88 110
 69  92 115
 72  96 120
 75 100 125
 80  84 116
```

```
100 N=100
110 FOR I = 1 TO N
120      FOR J = I+1 TO N
130        S=I*I+J*J
140        K=J
150        K=K+1
160        K2=K*K
170        IF K2 < S THEN 150
180        IF K2 > S THEN 200
190        PRINT USING 220,I,J,K
200        NEXT J
210    NEXT I
220 :###    ###    ###
230 END
```

Figura 2.5

Il listing di Fig.2.5 corrisponde ad un programma la cui uscita è molto "lineare" quindi un pò ingombrante. Per ridurre questa lunghezza si può preparare un'"edizione" su tre colonne. Per questo si utilizza una variabile ausiliaria B che fà eseguire un passaggio alla linea seguente dopo la stampa successiva di tre soluzioni. Abbiamo dunque modificato leggermente il flow-chart precedente nel seguente modo:

Questa soluzione porta ai listati di Figg.2.6 e 2.7

```
100 N=100
105 B-1
110 FOR I = 1 TO N
120   FOR J = I+1 TO N
130     S=I*I+J*J
140     K=J
150     K=K+1
160     K2=K*K
170     IF K2 < S THEN 150
180     IF K2 > S THEN 200
190     PRINT USING 220,I,J,K,
195     B=B+1
197     IF B <=3 THEN 200
198       PRINT 199
199       B=1
200     NEXT J
210   NEXT I
220 :###    ###    ###    I
230 END
```

Figura 2.6

Esempio di esecuzione:

```
  3    4     5 I  5    12    13 I  6     8    10 I
  7   24    25 I  8    15    17 I  9    12    15 I
  9   40    41 I 10    24    26 I 11    60    61 I
 12   16    20 I 12    35    37 I 13    84    85 I
 14   48    50 I 15    20    25 I 15    36    39 I
 16   30    34 I 16    63    65 I 18    24    30 I
 18   80    82 I 20    21    29 I 20    48    52 I
 20   99   101 I 21    28    35 I 21    72    75 I
 24   32    40 I 24    45    51 I 24    70    74 I
 25   60    65 I 27    36    45 I 28    45    53 I
 28   96   100 I 30    40    50 I 30    72    78 I
 32   60    68 I 33    44    55 I 33    56    65 I
 35   84    91 I 36    48    60 I 36    77    85 I
 39   52    65 I 39    80    89 I 40    42    58 I
 40   75    85 I 40    96   104 I 42    56    70 I
 45   60    75 I 48    55    73 I 48    64    80 I
 48   90   102 I 51    68    85 I 54    72    90 I
 56   90   106 I 57    76    95 I 60    63    87 I
 60   80   100 I 60    91   109 I 63    84   105 I
 65   72    97 I 66    88   110 I 69    92   115 I
 72   96   120 I 75   100   125 I 80    84   116 I
```

Figura 2.7

2.2 NUMERI DI ARMSTRONG

Si chiamano numeri di Armstrong, quei numeri che sono uguali alla somma dei cubi delle cifre che li compongono. Per esempio 153 è un numero di Armstrong poichè:

$$153 = 1^3 + 5^3 + 3^3$$

Scrivere un programma per ottenere tutti i numeri di Armstrong compresi tra 1 e 2000.

Analisi di un problema: Per analizzare ogni numero, si dovrà scomporlo in ognuna delle cifre che lo compongono ed in seguito calcolare la somma dei cubi delle cifre.

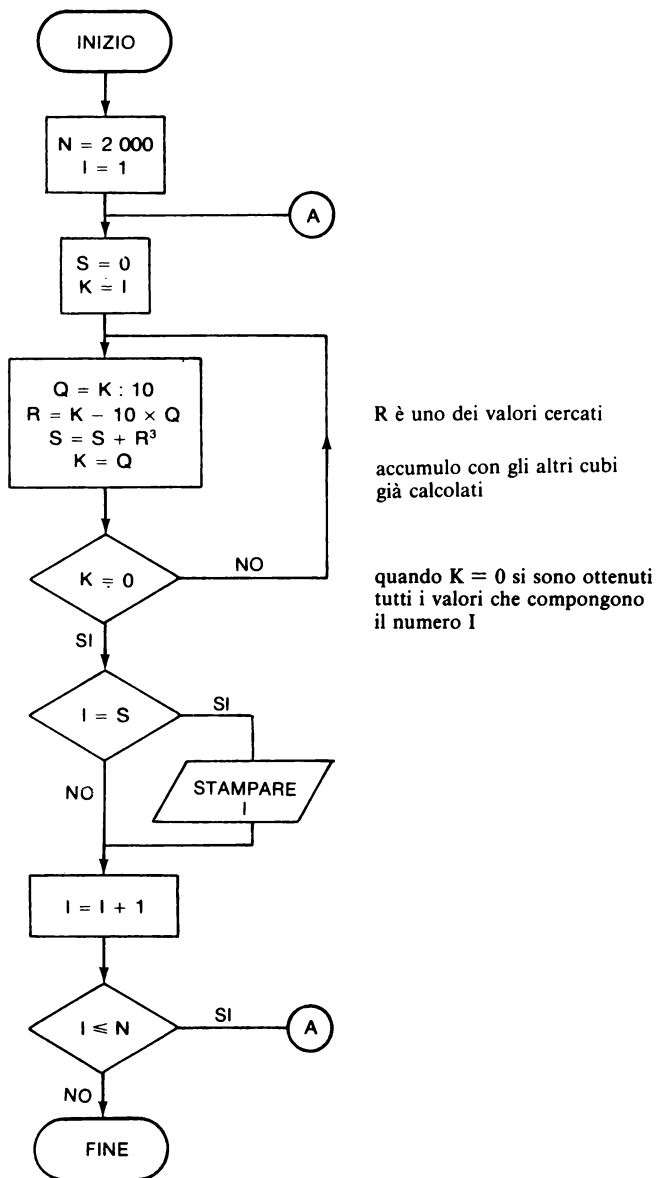



Figura 2.8

Per ottenere la cifra delle unità, è sufficiente calcolare il resto della divisione intera di questo numero per 10. Se I è il numero, calcoleremo come segue:

Q = INT (I/10)
R = I - 10 x Q   R dà la cifra delle unità

Per ottenere ora la cifra delle decine, sarà sufficiente ricominciare la stessa operazione partendo da Q.

Q = INT (Q/10)
R = Q - 10 · Q1

Un modo comune di procedere può essere quello di continuare finchè il quoziente della divisione è uguale a zero. Se si limita N = 2000, non si supereranno le quattro cifre.

Piuttosto che calcolare Q1 poi Q2 e così di seguito, si può operare nel seguente modo:

1. Porre K = I
2. Calcolare Q = intero (K/10)
   R = K − 10 x Q
   porre S = S + R3
   porre K = Q per la seguente iterazione
   se K > 0 ritornare in 2 altrimenti andare in 3
3. Provare se S = I

Questo ci porta al flow-chart di Fig. 2.8.

Il listato di Fig. 2.9 corrisponde al flow-chart di Fig. 2.8. L'esempio di esecuzione mostra che i numeri di Armstrong non sono poi tanti.

```
10  N=2000
20  PRINT"NUMERI DI ARMSTRONG TRA 1 E 2000 "
30  PRINT
40  FOR I=1 TO N
50    S=0
60    K=I
70    Q=INT(K/10)
80    R=K-10*Q
90    S=S+R*R*R
100    K=Q
110    IF K<>0 THEN 70
120    IF I<>S THEN 130
125    PRINT I
130    NEXT I
140 END
```

Figura 2.9

Esempio di esecuzione:

```
NUMERI DI ARMSTRONG TRA 1 E 2000
1
153
370
371
407
```

Figura 2.10

2.3 SCOMPOSIZIONE DI UNA FRAZIONE IN FRAZIONI EGIZIANE

Una frazione quando ha per numeratore 1 è chiamata frazione EGIZIANA (1), esempio 1/3, 1/10.

Si chiamano frazioni proprie quelle che hanno il numeratore più piccolo del denominatore.

Problema: scomporre una frazione propria in una somma di frazioni Egiziane. Per far questo proponiamo di utilizzare l'algoritmo di FIBONACCI (2).

Sia a/b la frazione da scomporre. Per determinare la prima frazione della scomposizione, si prende la più grande frazione Egiziana il cui valore è inferiore ad $a/10$. Si sottrae questa frazione da a/b e si continua con questa procedura fintanto che il resto sia uguale a zero.

Esempio:

$$a = 2 \quad b = 3 \quad \frac{a}{b} = \frac{2}{3}$$

la più grande frazione da prendere è $\frac{1}{2}$

$$\frac{a}{b} - \frac{1}{2} = \frac{2}{3} - \frac{1}{2} = \frac{1}{6}$$

dunque

$$\frac{2}{3} = \frac{1}{2} + \frac{1}{6}$$

la scomposizione non è sempre così rapida. Ad esempio per 8/11 si trova

$$\frac{8}{11} = \frac{1}{2} + \frac{1}{5} + \frac{1}{55} + \frac{1}{110}$$

Esercizio: Costruire un programma che scomponga una frazione in frazioni Egiziane secondo l'algoritmo di Fibonacci. Porre particolare attenzione alla rappresentazione dei risultati. Si discuterà inoltre circa i limiti del programma e le precauzioni da prendere.

---

(1) Queste frazioni erano utilizzate nell'antichità dagli Egiziani che non riuscivano ad utilizzare altri tipi di frazioni.

(2) FIBONACCI: Leonardo da PISA conosciuto sotto il nome di Fibonacci nato verso il 1175 a Pisa, pubblicò questo algoritmo nel 1202.

Soluzione: Questo problema è in apparenza molto semplice, è necessario:
- cercare la più grande frazione Egiziana inferiore ad *a/b*
- calcolare la frazione restante.

Se si parte da C = intero di B/A, si avrà,

$$\frac{1}{C} \geqslant \frac{A}{B}$$

ma C sarà molto vicino al denominatore cercato. E' sufficiente dunque porre C = C+ 1 fino a che $1/c < A/B$ e si è allora trovata la frazione cercata.

La frazione restante si ottiene da:

$$\frac{A}{B} - \frac{1}{C}$$

che è dato come $\begin{cases} AC - B = \text{numeratore} \\ BC = \text{denominatore} \end{cases}$

Partendo da questa analisi si può disegnare un primo flow-chart. Ma si può già fare un'importante considerazione: i calcoli devono essere effettuati unicamente con numeri interi che possono diventare molto grandi. E' noto che la rappresentazione dei numeri in BASIC dà una precisione limitata. E' necessario dunque prevedere un test per assicurarsi che non sia stata superata la precisione del calcolatore utilizzato.

Questo test dovrà essere fatto a priori dopo ogni moltiplicazione AxC e BxC, ma quando $B > A$, è sufficiente fare un test sulla seconda moltiplicazione. Questo porta al flow-chart delle Figg. 2.11 e 2.12. Il test di arresto è $B < P$.

NOTA: Come nei problemi precedenti, si possono utilizzare le variabili intere se il computer utilizzato lo permette. Tuttavia per i microcalcolatori questo porta, generalmente, ad una capacità inferiore. E' dunque preferibile lavorare con delle variabili normali.

Il programma è stato realizzato in due parti:
- un programma principale che effettua l'I/O e qualche test,
- un sottoprogramma che a ciascuna "iterazione" cerca la più grande frazione Egiziana possibile, e la frazione restante per la tappa successiva.

Spezzando il programma in due parti, si aumenta un pò il numero delle istruzioni, ma la sua scrittura è resa più semplice.

Nel flow-chart di Fig. 2.11, si fà appello alla variabile L che assume solo 2 valori: 0 e 1. Inizialmente questa variabile è posta a zero e permette, grazie al test, di non far precedere le prime frazioni trovate dal segno +. Poi, assume il valore 1 e ogni frazione seguente è preceduta dal segno +. Si ottiene così una stampa molto "elegante" sebbene tutto sia sistemato su una sola linea.

INIZIO
LEGGERE PRECISIONE
LEGGERE A E B
SI
A ≥ B
FINE
L=0
STAMPARE FRAZIONE A/B =
SI
A=0
NO
SOTTOPROGRAMMA DI SCOMPOSIZIONE
NO
L=0
SI
STAMPA +
STAMPA LA FRAZIONE EGIZIANA OTTENUTA 1/C
L=1
SI
B < P
NO
CODICE ERRORE

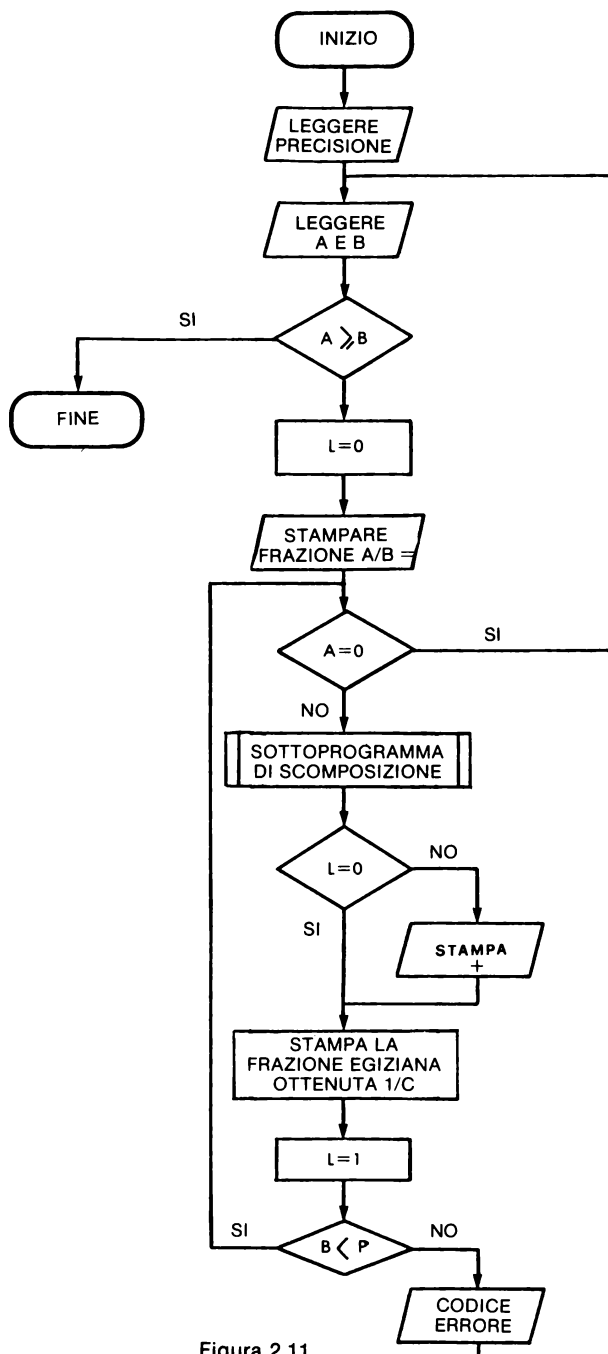
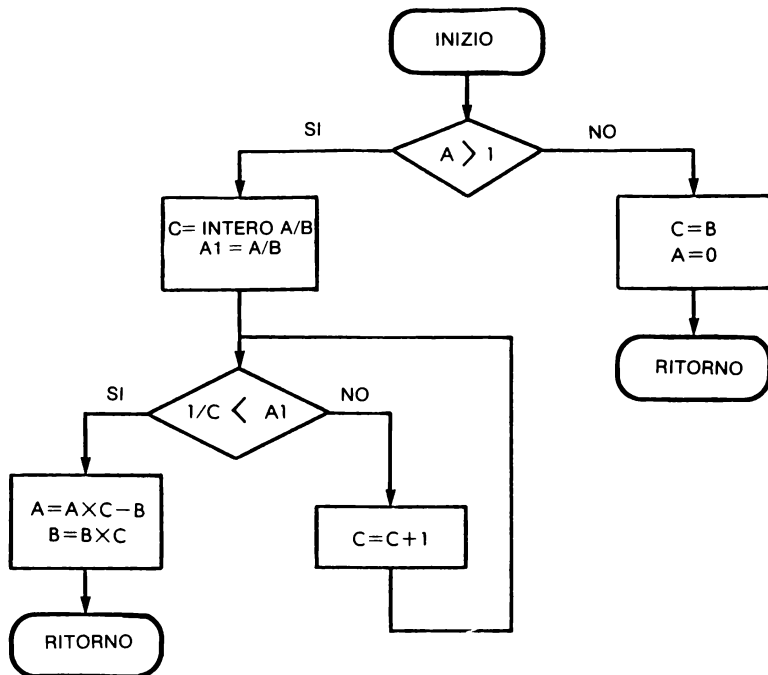



Figura 2.11

Figura 2.12

Commenti al programma:

- Per un calcolatore la precisione dei dati è costante. Si può dunque:

- sia indicare al programma il numero più grande possibile nel programma per mezzo di un'istruzione di assegnazione oppure delle istruzioni READ e DATA. Quest'ultima soluzione è quella adottata,

- sia domandare all'utilizzatore di indicare il numero più grande ammissibile durante l'esecuzione. Questa soluzione è meno pratica da realizzare.

- Per fermare il programma è sufficiente introdurre due numeri A e B in modo che A sia maggiore di B.

Suggerimenti: Partendo da questo programma se ne può scrivere un'altra versione di tipo "conversazionale" che permetterà all'utilizzatore stesso di provare un diversa scomposizione, senza che sia obbligato ad effettuare i calcoli. Il programma s'incarica di indicare la frazione restante.

```
100  PRINT "SCOMPOSIZIONE IN FRAZIONI EGIZIANE"
105  READ P
110  PRINT
120  PRINT
130  PRINT "IMPOSTARE NUMERATORE, DENOMINATORE  ";
140  INPUT A,B
150  IF A >  = B THEN 800
160 L = 0
170  PRINT : PRINT
180  PRINT "FRAZIONE ";A;"/";B;" = ";
190  IF A = 0 THEN 110
200  GOSUB 500
210  IF L = 0 THEN 230
220  PRINT " + ";
230  PRINT "1/";C;
235 L = 1
240  IF B < P THEN 190
300  PRINT "  SUPERO DI CAPACITA'"
310  GOTO 110
500  IF A > 1 THEN 600
510 C = B
520 A = 0
530  RETURN
600 C =  INT (B / A)
605 A1 = A / B
610  IF 1 / C < A1 THEN 640
620 C = C + 1
630  GOTO 610
640 A = A * C - B
650 B = B * C
670  RETURN
700  DATA  32767      <———— massimo numero intero ammissibile
800  END
```

Figura 2.13

Esempio di esecuzione:

```
SCOMPOSIZIONE IN FRAZIONI EGIZIANE

IMPOSTARE NUMERATORE, DENOMINATORE  ?2,3

FRAZIONE 2/3 = 1/2 + 1/6
IMPOSTARE NUMERATORE, DENOMINATORE  ?8,11

FRAZIONE 8/11 = 1/2 + 1/5 + 1/37 + 1/4070
IMPOSTARE NUMERATORE, DENOMINATORE  ?3,7

FRAZIONE 3/7 = 1/3 + 1/11 + 1/231
IMPOSTARE NUMERATORE, DENOMINATORE  ?7,13

FRAZIONE 7/13 = 1/2 + 1/26
IMPOSTARE NUMERATORE, DENOMINATORE  ?13,29

FRAZIONE 13/29 = 1/3 + 1/9 + 1/262  SUPERO DI CAPACITA'

IMPOSTARE NUMERATORE, DENOMINATORE  ?11,29

FRAZIONE 11/29 = 1/3 + 1/22 + 1/1914

IMPOSTARE NUMERATORE, DENOMINATORE  ?3,2
```

Figura 2.14

2.4 I NUMERI PRIMI

Un primo metodo per trovare i numeri primi consiste nel cercare, partendo dal numero 3, i numeri dispari che accettano come divisori solo se stessi o la cifra 1.

Studieremo in seguito un metodo più raffinato.

2.4.1 Primo metodo: Scrivere un semplice programma che stampi N numeri primi. Per l'esecuzione si prenderà N tra 10 e 60. Studiare poi come migliorare la presentazione.

2.4.2 Soluzione: L'ossatura generale del programma corrisponde al flow-chart di Fig. 2.15.

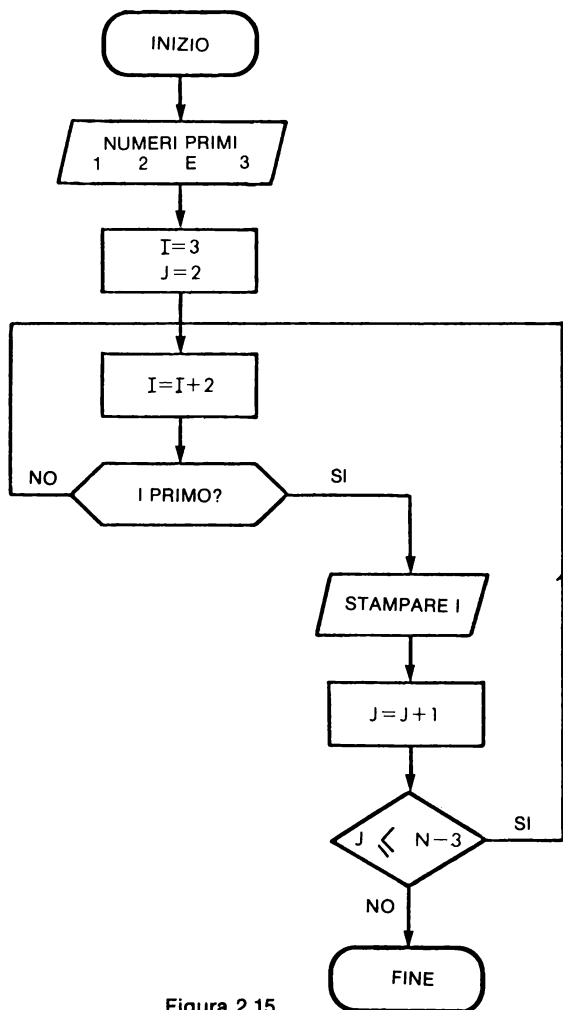



Figura 2.15

— stampare i numeri 1, 2 e 3
— poi cercare, in seguito, N-3 altri numeri primi facendo avanzare I di due passi per volta; di due passi perchè i numeri primi sono tutti dispari.

Per determinare se I è un numero primo, sarà necessario eseguire test successivi utilizzando numeri dispari fino ad ottenere uno dei seguenti casi:

— si ottiene come resto 0, questo significa che I non è un numero primo
— si ottiene come resto un numero diverso da zero ed un quoziente inferiore o uguale al divisore, in questo caso I è un numero primo.

Per rispondere alla domanda "I è un numero primo ?", è necessario eseguire una parte del flow-chart di Fig. 2.16. Si possono ora tracciare i dettagli del flow-chart e soprattutto si può scrivere il programma.

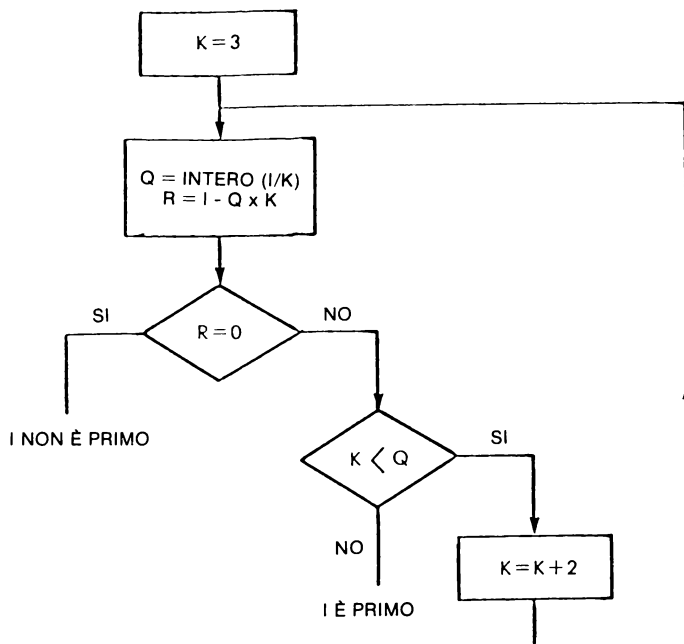



Figura 2.16

```
100 N = 60
110  PR# 1: PRINT "                "
115  PRINT "I PRIMI ";N;" NUMERI PRIMI SONO: "
120  PRINT 1,2,3
130 I = 3
140  FOR J = 1 TO N - 3
150 I = I + 2
160 K = 3
170 Q =  INT (I / K)
180 R = I - Q * K
190  IF R = 0 THEN  GOTO 150
200  IF K > Q THEN  GOTO 230
210 K = K + 2
220  GOTO 170
230  PRINT I,
240  NEXT J
250  END
```

Figura 2.17

Esempio di esecuzione:

```
I PRIMI 60 NUMERI PRIMI SONO:
1              2              3
5              7              11             13             17
19             23             29             31             37
41             43             47             53             59
61             67             71             73             79
83             89             97             101            103
107            109            113            127            131
137            139            149            151            157
163            167            173            179            181
191            193            197            199            211
223            227            229            233            239
241            251            257            263            269
271            277
```

Figura 2.18

2.4.3 Secondo metodo: Cominciando dal numero 5 incluso, tutti i numeri primi sono dati come $6\,n + 1$, con n intero. Inoltre è sufficiente cercare i divisori tra i numeri primi già trovati. Scrivere un programma che tenga conto di queste due note.

2.4.4 Soluzione: Per ricercare gli eventuali divisori fra i numeri primi già trovati è necessario poterli memorizzare, questo rende necessario l'utilizzo di una tabella la cui dimensione limiterà il numero massimo dei numeri primi che si potranno cercare. Inoltre i numeri dati come $6\,n + 1$ sono indivisibili per 2 e per 3, sarà quindi sufficiente cercare dei divisori a partire dal numero 5.

Scomponiamo il lavoro in due parti:

— un programma principale inizializza la tabella T, aggiorna la variabile A e chiama un sottoprogramma.

— il sottoprogramma prova se la variabile A rappresenta un numero primo, se si la pone nella tabella T. Quando la tabella T è riempita il contenuto viene stampato.

Si giunge così ai flow-chart di Figg. 2.19 e 2.20.

Esempio di esecuzione:

2.5 SCOMPOSIZIONE IN FATTORI PRIMI

Scomporre un numero in fattori primi consiste nel cercare tutti i divisori primi di questo numero.

2.5.1 Metodo elementare: Si cercheranno i divisori a partire dal numero 2. Dal momento in cui si trova un divisore lo si stampa. Quando un divisore non è più conveniente, si passa al divisore successivo.

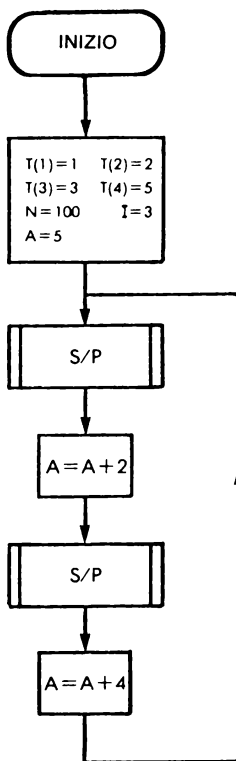



Figura 2.19

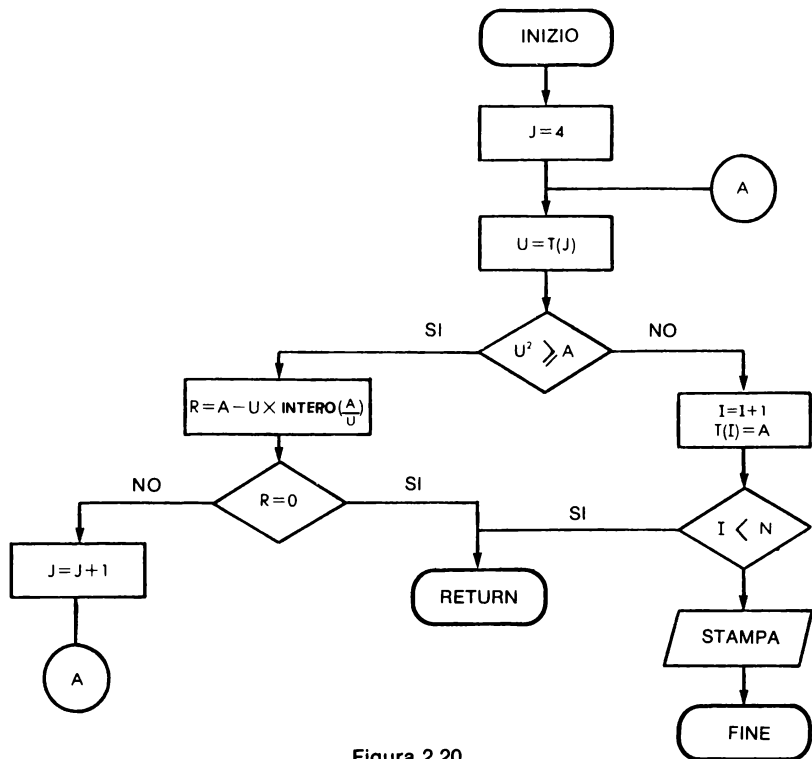



Figura 2.20

```
100  DIM T(200)
110 T(1) = 1:T(2) = 2:T(3) = 3:T(4) = 5
150 B = 5:I = 3
160 N = 200
170  GOSUB 500
180 B = B + 2
190  GOSUB 500
200 B = B + 4
210  GOTO 170
500  FOR J = 4 TO I
510 U = T(J)
520  IF U * U < B THEN 560
530 R = B -  INT (B / U) * U
540  IF R = 0 THEN  RETURN
550  NEXT J
560 I = I + 1
570 T(I) = B
580  IF I < N THEN  RETURN
590  PRINT "IL SEGUENTE LISTATO CONTIENE ";N;" NUMERI PRIMI:"
600  PRINT
610 K = 0
620  FOR I = 1 TO N
630  PRINT  TAB( 7 * K);T(I);
640 K = K + 1
650  IF K > 10 THEN 670
660 K = 0: PRINT
670  NEXT I
680  END
```

Figura 2.21

Esempio di esecuzione:

IL LISTATO CONTIENE 200 NUMERI PRIMI

1	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	1009
1013	1019	1021	1031	1033	1039	1049	1051	1061	1063
1069	1087	1091	1093	1097	1103	1109	1117	1123	1129
1151	1153	1163	1171	1181	1187	1193	1201	1213	1217

Figura 2.22

Se, ad un certo punto, si trova un quoziente più piccolo del divisore:

— se il dividendo è il numero dato, questo è il numero primo,
— se il dividendo è inferiore, questo allora è un numero primo che divide il numero.

Costruire un programma che effettua questa scomposizione, che poi richieda un altro numero finchè non gli venga dato un numero negativo oppure zero, il che costituisce un criterio di arresto.

2.5.2 Soluzione: La struttura generale del programma è data dal flow-chart di Fig.

2.23. Bisogna ora dettagliare la parte "scomposizione".

Per effettuare questa scomposizione si può utilizzare il seguente algoritmo:

1. Ricopiare N in N_1

2. Per I che assume successivamente i valori 2, 3, 4, 5, etc.

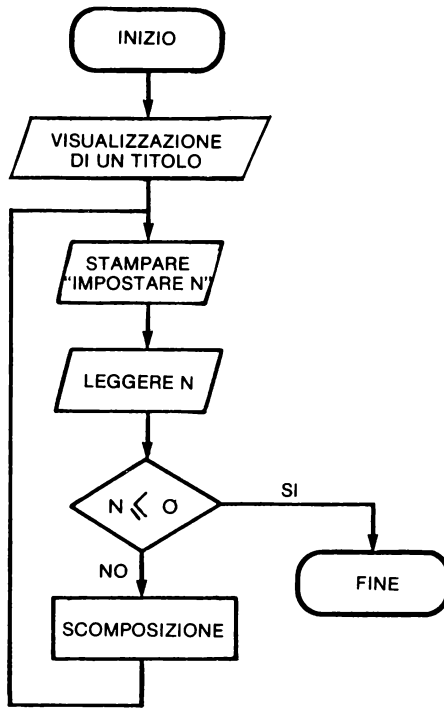



Figura 2.23

2.a Provare se I è un divisore di N,
sia Q il quoziente,
se I è un divisore, stampare I, porre N=Q e ritornare in 2.a,
se I non è un divisore andare in 2.b.

2.b Se il quoziente Q è più grande di I, incrementare I ritornando in 2,
se il quoziente Q è ≤ I allora:

se N=1 è terminata la scomposizione
se $N=N_1$ allora N è primo,
se $N<N_1$ allora N è un divisore e bisogna stamparlo alla fine della scomposizione.

Questo metodo porta al flow-chart di Fig. 2.24 partendo dal quale si è scritto, senza difficoltà, il programma in Fig. 2.25.

N1 = N
I = 1
sottoprogramma
di scomposizione
I = I + 1
Q = INTERO (N/1)
R = N — QX1
NO
R = 0
SI
SI
Q > I
N = Q
NON
STAMPARE I
N = 1
OUI
NO
N = N1
NO
SI
STAMPARE
È PRIMO
STAMPARE N
FINE
SCOMPOSIZIONE

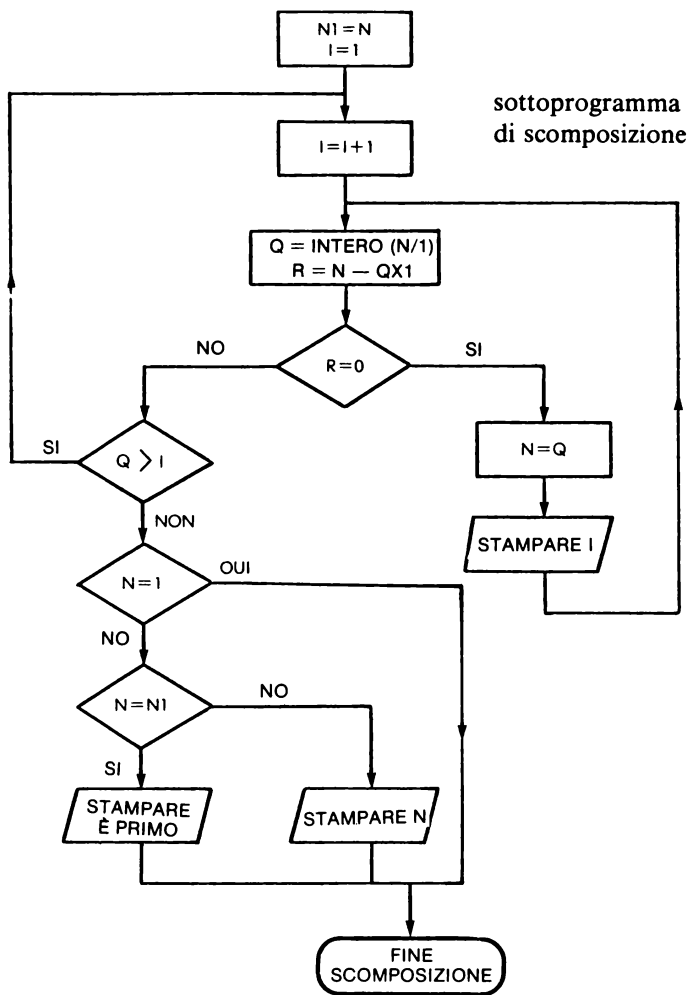



Figura 2.24

```
100  REM  SCOMPOSIZIONE IN FATTORI PRIMI
110  REM
120  PRINT "SCOMPOSIZIONE IN FATTORI PRIMI"
130  PRINT : PRINT
140  INPUT "IMPOSTARE IL NUMERO DA SCOMPORRE  ";N
150  IF N < = 0 THEN  STOP
160 N1 = N
170 I = 1
180 I = I + 1
200 Q =  INT (N / I)
210 R = N - Q * I
220  IF R < > 0 THEN 290
230 N = Q
240  PRINT I" ";
250  GOTO 200
290  IF Q > I THEN 180
300  IF N = 1 THEN 350
310  IF N < > N1 THEN 340
320  PRINT  TAB( 5);"E' PRIMO."
330  GOTO 350
340  PRINT N
350  PRINT
360  GOTO 130
370  END
```

Figura 2.25

Esempio di esecuzione:

```
SCOMPOSIZIONE IN FATTORI PRIMI

IMPOSTARE IL NUMERO DA SCOMPORRE  12
2 2 3

IMPOSTARE IL NUMERO DA SCOMPORRE  38
2 19

IMPOSTARE IL NUMERO DA SCOMPORRE  262144
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

IMPOSTARE IL NUMERO DA SCOMPORRE  65784
2 2 2 3 2741

IMPOSTARE IL NUMERO DA SCOMPORRE  1217
    E' PRIMO.

IMPOSTARE IL NUMERO DA SCOMPORRE  35427
3 7 7 241
IMPOSTARE IL NUMERO DA SCOMPORRE  0

BREAK IN 150
```

Figura 2.26

2.5.3 Metodo per una elegante presentazione: La presentazione dei risultati ottenuti con il metodo precedente non è certo molto gradevole. Migliorarla per ottenere il seguente risultato:

```
SCOMPOSIZIONE IN FATTORI PRIMI


IMPOSTARE IL NUMERO DA SCOMPORRE        65784
   E' DIVISIBILE PER 2           3                    VOLTE
   E' DIVISIBILE PER 3           1                    VOLTE
   E' DIVISIBILE PER 2741        1                    VOLTA
```

```
IMPOSTARE IL NUMERO DA SCOMPORRE        1217
    E' PRIMO.

IMPOSTARE IL NUMERO DA SCOMPORRE        35427
   E' DIVISIBILE PER 3          1                   VOLTE
   E' DIVISIBILE PER 7          2                   VOLTE
   E' DIVISIBILE PER 241        1                   VOLTA

IMPOSTARE IL NUMERO DA SCOMPORRE        262144
   E' DIVISIBILE PER 2          18                  VOLTE

IMPOSTARE IL NUMERO DA SCOMPORRE        19
    E' PRIMO.

IMPOSTARE IL NUMERO DA SCOMPORRE        14
   E' DIVISIBILE PER 2          1                   VOLTE
   E' DIVISIBILE PER 7          1                   VOLTA

IMPOSTARE IL NUMERO DA SCOMPORRE        0

BREAK IN 160
```

Figura 2.27

Si chiede di costruire il programma corrispondente.

2.5.4 Soluzione: Si può partire dal flow-chart di Fig. 2.24 che dovrà essere modificato per stampare solo quando si termina con un divisore. La parte incorniciata del flow-chart precedente è dunque sostituita dal flow-chart che segue:

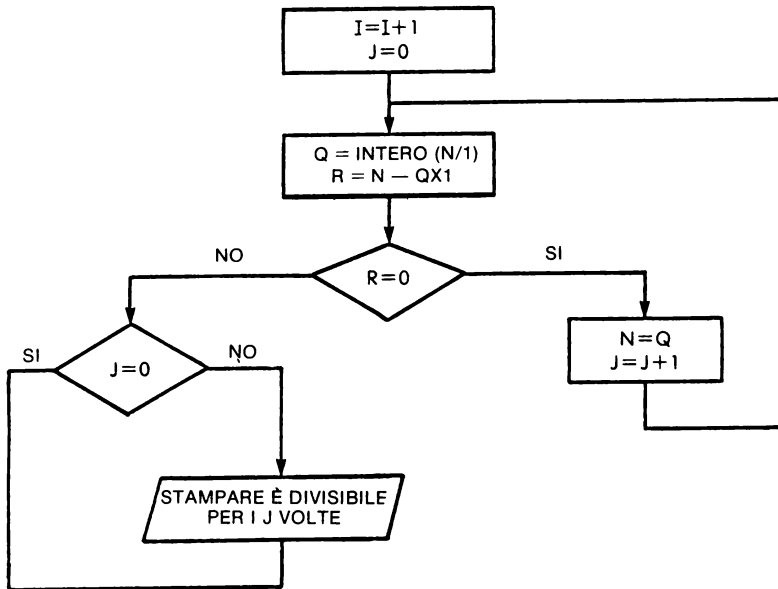



Figura 2.28

E' ora sufficiente modificare il programma precedente per giungere alla nuova stesura. Oltre alle modifiche date per la parte di flow-chart, non bisognerà dimenticare di modificare le istruzioni di stampa. Questo porta al listato di Fig. 2.29.

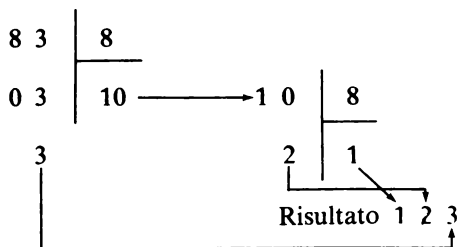
```
100  REM  SCOMPOSIZIONE IN FATTORI PRIMI
110  REM
120  PRINT "SCOMPOSIZIONE IN FATTORI PRIMI"
130  PRINT : PRINT
140  INPUT "IMPOSTARE IL NUMERO DA SCOMPORRE  ";N
150 N1 = N
160  IF N <  = 0 THEN  STOP
170 I = 1
180 I = I + 1
190 J = 0
200 Q =  INT (N / I)
210 R = N - Q * I
220  IF R <  > 0 THEN 260
230 N = Q
240 J = J + 1
250  GOTO 200
260  IF J = 0 THEN 290
270  PRINT "   E' DIVISIBILE PER ";I;" "; TAB( 10);J; TAB( 18);" (C
280  GOTO 180
290  IF Q > I THEN 180
300  IF N = 1 THEN 350
310  IF N <  > N1 THEN 340
320  PRINT  TAB( 5);"E' PRIMO."
330  GOTO 350
340  PRINT "   E' DIVISIBILE PER ";N; TAB( 10);" 1"; TAB( 18);" (OL(A
350  PRINT
360  GOTO 140
370  END
```

Figura 2.29

2.6 CONVERSIONE BASE 10 IN UN'ALTRA BASE

La rappresentazione di un numero in differenti sistemi di numerazione (base 10, base 8, base 2, etc.) costituisce per "l'uomo della strada", un esercizio matematico senza concreta utilità. Invece, per gli "informatici", questo lavoro presenta un'utilità reale soprattutto quando si programma in linguaggio assembler. Per questa ragione sarà presentato questo esercizio. Il principio della conversione è il seguente:

— effettuare delle divisioni successive per la nuova base fino a che non si ottenga un quoziente inferiore al valore della nuova base;

Esempio: conversione di 83 scritto in base 10 nella base 8:

— la cifra delle unità corrisponde al primo resto. La cifra seguente corrisponde al resto seguente e così via. L'ultima cifra è l'ultimo quoziente.

Così 83 dà in base 8 123,
83 dà in base 7 146.

2.6.1 Conversione in base minore di dieci: Scrivere un programma che stampi una tabella di conversione tra due numeri D e F dati dall'utilizzatore. La conversione si effettua da base 10 verso un'altra base inferiore a 10.

Soluzione: La costruzione di questo programma ha qualcosa di familiare con i programmi precedenti:

— utilizzo della "divisione intera",
— calcolo del resto,
— inserimento in una tabella.

Per meglio distinguere la struttura generale dell'algoritmo propriamente detto, si scomporrà il programma in due parti: il programma principale che effettua l'I/O e un sottoprogramma corrispondente all'algoritmo di cambiamento di base.

La costruzione del flow-chart di Fig. 2.30 non presenta difficoltà. Mentre il flow-chart di Fig. 2.31 necessita di alcune spiegazioni.

— Quando si comincia ad effettuare la conversione, non si sa a priori quante cifre significative avrà il nuovo numero: sarà dunque obbligatorio inserire le cifre nell'ordine in cui si ottengono.

— Il metodo proposto fornisce prima le cifre relative alle unità, poi quelle delle decine (o più esettamente quelle corrispondenti a un esponente 1) e così di seguito. Per inserirle nella tabella A si procede partendo da J=1.

A ogni cifra trovata A (J) = R
J = J + 1 (per la successiva cifra da inserire)

Per l'ultima cifra si pone A(J) = Q

Si può ora costruire il flow-chart di Fig. 2.31 del sottoprogramma di conversione e in seguito scrivere il programma completo. Per la stampa del numero desiderato, è necessario procedere nell'ordine inverso rispetto a quello utilizzato per ottenerlo; per esempio se il numero convertito è 127, si è ottenuto:

A (1) = 7
A (2) = 2
A (3) = 1 in questo caso L = 3

Per stampare con ordine si scriverà:

```
FOR I=L TO 1 STEP —1
PRINT A(I);              per restare sulla stessa linea
NEXT I
PRINT                    per passare alla linea seguente
```

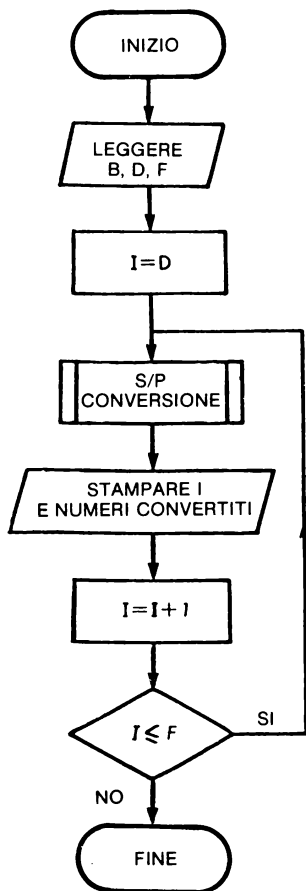



Figura 2.30

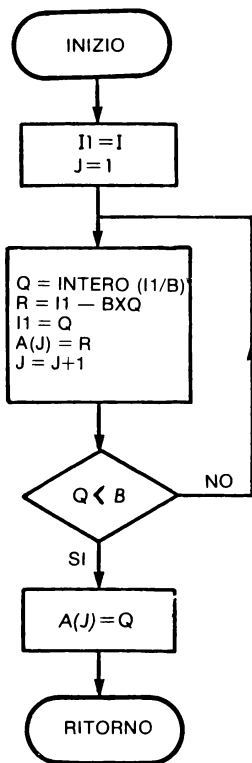



Figura 2.31

```
95  DIM A(15)
100  PRINT "LA NUOVA BASE ";
110  INPUT B
120  PRINT "PRIMO E ULTIMO NUMERO DA CONVERTIRE ";
130  INPUT F,L
140  FOR I = F TO L
150  PRINT
160  GOSUB 1500
170  REM  STAMPA LA TABELLA
180  PRINT I; TAB( 6);
190  FOR D = J TO 1 STEP  - 1
200  PRINT A(D);
210  NEXT D
220  NEXT I
230  STOP
1500 I1 = I
1510 J = 1
1520 Q =  INT (I1 / B)
1530 R = I1 - Q * B
1535 I1 = Q
1540 A(J) = R
1545 J = J + 1
1550  IF Q >  = B THEN 1520
1560 A(J) = Q
1570  RETURN
1580  END
```

```
LA NUOVA BASE ?16
PRIMO E ULTIMO NUMERO DA
CONVERTIRE ?16380,16390
16380      3151512
16381      3151513
16382      3151514
16383      3151515
16384      4000
16385      4001
16386      4002
16387      4003
16388      4004
16389      4005
16390      4006
BREAK IN 230
]
```

Figura 2.32

Esempio di esecuzioni.

```
LA NUOVA BASE ?2
PRIMO E ULTIMO NUMERO DA CONVERTIRE ?260,280
260      100000100
261      100000101
262      100000110
263      100000111
264      100001000
265      100001001
266      100001010
267      100001011
268      100001100
269      100001101
270      100001110
271      100001111
272      100010000
273      100010001
274      100010010
275      100010011
276      100010100
277      100010101
278      100010110
279      100010111
280      100011000
BREAK IN 230
]
```

```
LA NUOVA BASE ?8
PRIMO E ULTIMO NUMERO DA
CONVERTIRE ?70,90
70      106
71      107
72      110
73      111
74      112
75      113
76      114
77      115
78      116
79      117
80      120
81      121
82      122
83      123
84      124
85      125
86      126
87      127
88      130
89      131
90      132
BREAK IN 230
]
```

Figura 2.33

2.6.2 Conversione in base maggiore di 10: Estendere il programma al fine di permettere la stampa di una tabella di conversione per una base maggiore di 10. In questo caso si rappresenterà la cifra 10 con la lettera A, 11 con la lettera B e così di seguito.

Soluzione: Si utilizzeranno le funzioni relative alle stringhe di caratteri; si può per esempio dare una stringa B$ tale che:

B$ = "0123456789ABCDEFGHIJKL"

A partire dall'A(L) precedente, è sufficiente estrarre il carattere in posizione A(L)+1 nella stringa (la cifra 0 corrisponde al primo carattere di B$).

Utilizzando, in funzione del sistema disponibile, SUBSTR o MID$, si scriverà:

```
00200  PRINT  SUBSTR (B$ , A (L) + 1 , 1);
```

per stampare il carattere appropriato.

Questo porta al programma della Fig. 2.34.

```
10  REM  PROGRAMMA DI CONVERSIONE
50  DIM A(15)
90 B$ = "0123456789ABCDEFGHIJKLMN"
100  PRINT "LA NUOVA BASE ";
110  INPUT B
120  PRINT "PRIMO E ULTIMO NUMERO DA CONVERTIRE ";
130  INPUT F,L
140  FOR I = F TO L
150  PRINT
160  GOSUB 1500
170  REM  STAMPA LA TABELLA
180  PRINT I; TAB( 6);
190  FOR D = J TO 1 STEP  - 1
200  PRINT  MID$ (B$,A(D) + 1,1);
210  NEXT D
220  NEXT I
230  STOP
1480  REM  SUBROUTINE DI CONVERSIONE
1500 I1 = I
1510 J = 1
1520 Q =  INT (I1 / B)
1530 R = I1 - Q * B
1535 I1 = Q
1540 A(J) = R
1545 J = J + 1
1550  IF Q >  = B THEN 1520
1560 A(J) = Q
1570  RETURN
1580  END
]
```

Figura 2.34

```
LA NUOVA BASE ?2
PRIMO E ULTIMO NUMERO
DA CONVERTIRE ?260,280
260       100000100
261       100000101
262       100000110
263       100000111
264       100001000
265       100001001
266       100001010
267       100001011
268       100001100
269       100001101
270       100001110
271       100001111
272       100010000
273       100010001
274       100010010
275       100010011
276       100010100
277       100010101
278       100010110
279       100010111
280       100011000
```

```
LA NUOVA BASE ?8
PRIMO E ULTIMO NUMERO DA CONVERTIRE ?75,90
75      113
76      114
77      115
78      116
79      117
80      120
81      121
82      122
83      123
84      124
85      125
86      126
87      127
88      130
89      131
90      132


LA NUOVA BASE ?16
PRIMO E ULTIMO NUMERO DA CONVERTIRE ?1023,1035
1023      3FF
1024      400
1025      401
1026      402
1027      403
1028      404
1029      405
1030      406
1031      407
1032      408
1033      409
1034      40A
1035      40B
BREAK IN 230
]
```

Figura 2.35

Casi particolari:

Alcuni calcolatori non dispongono delle funzioni SUBSTR o MID$ ma esiste comunque un'altra possibilità: dopo aver lanciato il programma dichiarare la lunghezza massima della variabile stringa di caratteri B$ si può estrarre una sottostringa scrivendo l' espressione:

B$ (I,J)

nella quale I rappresenta la posizione del primo carattere della sottostringa e J la posizione dell'ultimo. E' il caso, per esempio, del calcolatore LOGABAX LX 500. Si scriverà dunque in questo caso:

```
A1 = A ( L ) + 1
PRINT B$ (A1, A1)
```

per stampare solo un carattere, da cui il programma di Fig. 2.36.

```
10  REM  CONVERSIONE DI BASE
20  REM  AUTORE J.P. LOMOITIER
30  DIM A(15),B$(30)
40 B$ = "0123456789ABCDEFGHIJKLMNOPQR"
100  INPUT "IMPOSTARE LA NUOVA BASE ",B
110  PRINT
120  INPUT "IMPOSTARE L'INIZIO E LA FINE DELLA TABELLA ",D,F
125  PRINT
130  PRINT "BASE 10", TAB( 10),"BASE ",B
140  FOR I = D TO F
150  PRINT
160  GOSUB 1500
170  REM  STAMPA
180  PRINT I, TAB( 11),
190  FOR L = J TO 1 STEP  - 1
195 A1 = A(L) + 1
200  PRINT B$(A1,A1),
210  NEXT L
220  NEXT I
230  STOP
1500 I1 = I
1510 J = 1
1520 Q =  INT (I1 / B)
1530 R = I1 - Q * B
1535 I1 = Q
1540 A(J) = R
1545 J = J + 1
1550  IF Q <  = B THEN 1520
1560 A(J) = Q
1570  RETURN
1580  END
```

```
IMPOSTARE LA NUOVA BASE 16
IMPOSTARE L'INIZIO E LA FINE
DELLA TABELLA 12,20

BASE 10    BASE 16

 12          0C
 13          0D
 14          0E
 15          0F
 16          10
 17          11
 18          12
 19          13
 20          14
```

Figura 2.36

2.7 CONCLUSIONE

Questi esercizi di differenti difficoltà mostrano l'utilità della costruzione di un flow-chart per gradi, partendo dalla struttura generale del programma per poi procedere dettagliatamente fino a sentirsi pronti per la scrittura dello stesso.

Per risolvere un problema, la soluzione in generale non è una sola sia sul piano del metodo che su quello della programmazione. I programmi presentati in questo libro non sono tra i migliori dal punto di vista dell'efficienza, ma sono in generale i più facili da comprendere e molto prossimi al flow-chart. Man mano che si acquisisce esperienza, si può ridurre il tempo destinato alla costruzione del flow-chart, contentandosi di un flow-chart di principio dal quale ricavare la scrittura del programma.

CAPITOLO 3

SEMPLICI ESERCIZI DI GEOMETRIA

3.0 INTRODUZIONE

La geometria classica non si presta ad applicazioni numeriche, al contrario della "geometria analitica" elementare. Gli esercizi presentati in questo libro sono di facile comprensione e mostrano le capacità del microcalcolatore. Sono stati scelti per il loro interesse pratico e per la loro semplicità: il flow-chart si costruisce facilmente e la programmazione è ugualmente facile. I calcoli, necessitando di una certa precisione, risultano complessi se eseguiti a mano mentre sono eseguiti molto rapidamente con l'ausilio del calcolatore.

Partendo da questi esempi il lettore accorto ne potrà realizzare altri più complessi o più adatti alle proprie esigenze.

3.1 PERIMETRO E AREA DI UN TRIANGOLO

Per calcolare la superficie di un triangolo qualunque, si può misurare la lunghezza di ogni lato e ottenere così l'area del triangolo applicando la formula di Erone.

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

con

$$s = \frac{a+b+c}{2}$$ essendo a, b e c le lunghezze dei lati

Problema: Scrivere un programma che a partire da a, b, c calcoli il perimetro e l'area del triangolo.

Soluzione: Partendo da a, b, c il calcolo non presenta difficoltà particolari: si calcola $P = a + b + c$ che viene stampato, in seguito si applica la formula di Erone avendo

precedentemente calcolato il semiperimetro. Nel programma di Fig. 3.1 P rappresenta prima il perimetro poi il semiperimetro.

```
10 PRINT "IMPOSTARE LE TRE LUNGHEZZE DEI LATI DEL TRIANGOLO";
20 INPUT A,B,C
30 P=A+B+C
40 PRINT "PERIMETRO =";P
45 PRINT
50 P=.5*P
60 S=SQR(P*(P-A)*(P-B)*(P-C))
70 PRINT "SUPERFICIE =";S
80 END
```

Figura 3.1

Commenti:

Linea 10. Il punto e virgola, come la virgola posta alla fine di una linea, impedisce il passaggio alla linea successiva; questo permette dunque l'impostazione di dati sulla stessa linea.

Linea 40. PRINT "PERIMETRO" =;P. Il punto e virgola fa si che la stampa del valore numerico della variabile P sia posta vicino al segno =.

Linea 45. La sola istruzione PRINT provoca un salto di linea. Il suo utilizzo permette dunque la "buona" presentazione dei risultati.

Linea 50. Quando il valore del perimetro è stampato, non è più necessario questo valore, si può dunque utilizzare la stessa variabile per "inserire" il valore del semiperimetro, valore che sarà necessario in seguito.

Critica del programma:

Se vengono introdotte lunghezze di lati corrispondenti a un triangolo impossibile, per esempio 10, 20, 40, il calcolatore non avrà alcuna possibilità di rilevare questo errore e di segnalarlo. La macchina tenterà di estrarre la radice quadrata di un numero negativo, tutto ciò porta, generalmente, a spiacevoli conseguenze.

```
10 PRINT "IMPOSTARE LE TRE LUNGHEZZE DEI LATI DEL TRIANGOLO";
20 INPUT A,B,C
30 P=A+B+C
40 PRINT "PERIMETRO =";P
```

```
45 PRINT
50 P=.5*P
54 P1=(P-A)*(P-B)*(P-C)
56 IF P1>=0 THEN 60
57 PRINT "IMPOSSIBILE"
58 STOP
60 S=SQR(P*(P-A)*(P-B)*(P-C))
70 PRINT "SUPERFICIE =";S
80 END
```

Figura 3.2

Per rimediare al problema sarà necessario aggiungere un controllo di validità dei dati: la lunghezza del lato maggiore non deve superare la somma delle lunghezze degli altri due. Si può dunque aggiungere un test per verificare questa condizione oppure provare che

$(s - a)(s - b)(s - c)$ è positivo

Questa prova porta al programma di Fig. 3.2

3.2 DETERMINAZIONE DI UN CERCHIO PASSANTE PER TRE PUNTI

Partendo dalle coordinate cartesiane di tre punti M_1, M_2 e M_3 determinare il cerchio che passa per questi tre punti, cioè trovare le coordinate del centro e la lunghezza del raggio.

Analisi matematica:

Siano (x_1, y_1), (x_2, y_2), (x_3, y_3) le rispettive coordinate di M_1, M_2 e M_3. La pendenza dell'angolo della linea che congiunge M_1 a M_2 è data dalla:

$$\frac{y_2 - y_1}{x_2 - x_1}$$

quindi la pendenza di una perpendicolare è data da $-\frac{x_2 - x_1}{y_2 - y_1}$

l'equazione della mediana del segmento $M_1 M_2$ è

$$y = \frac{y_1 + y_2}{2} - \frac{x_2 - x_1}{y_2 - y_1}\left(x - \frac{x_1 + x_2}{2}\right)$$

dalla stessa equazione della mediana del segmento $M_1 M_3$ è

$$y = \frac{y_1 + y_3}{2} - \frac{x_3 - x_1}{y_3 - y_1} \left(x - \frac{x_1 + x_3}{2}\right)$$

Queste due equazioni si possono scrivere sotto la forma:

$$y = K_2 x + H_2$$
$$y = K_3 x + H_3$$

Con $K_2 - \frac{x_2 - x_1}{y_2 - y_1}$ $\qquad k_3 = - \frac{x_3 - x_1}{y_3 - y_1}$

$$H_2 = \frac{y_1 + y_2}{2} + \frac{x^2_2 - x^2_1}{2(y_2 - y_1)} \quad H_3 = \frac{y_1 + y_3}{2} + \frac{x^2_3 - x^2_1}{2(y_2 - y_1)}$$

Risolvendo il sistema lineare si ottengono le coordinate (X_0, Y_0) del centro I:

$$X_0 = \frac{H_3 - H_2}{K_2 - K_3}$$

$$Y_0 = \frac{K_3 H_2 - K_2 H_3}{K_3 - K_2}$$

Partendo dalle coordinate (X_0, Y_0) del centro I si ottiene la lunghezza R del raggio:

$$R = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

Flow chart:

La costruzione è realizzata senza difficoltà poichè è sufficiente seguire l'andamento del calcolo.

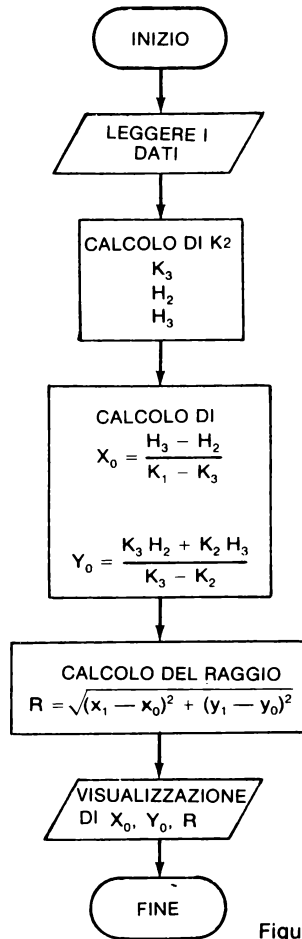



Figura 3.3

```
100 PRINT "DETERMINAZIONE DI UN CERCHIO PASSANTE PER TRE PUNTI"
110 PRINT
120 REM LE COORDINATE DEI TRE PUNTI DEVONO ESSERE PIAZZATE IN
121 REM UNA ISTRUZIONE 'DATA' PRIMA DELL'ESECUZIONE.
122 REM
130 READ X1,Y1,X2,Y2,X3,Y3
140 K2=-(X2-X1)/(Y2-Y1)
150 K3=-(X3-X1)/(Y3-Y1)
155 D=K3-K2
160 IF D=0 THEN 230
170 H2=0.5*(Y1+Y2+(X2*X2-X1*X1)/(Y2-Y1))
180 H3=0.5*(Y1+Y3+(X3*X3-X1*X1)/(Y3-Y1))
190 X0=(H2-H3)/D
200 Y0=(K3*H2-K2*H3)/D
```

```
210 R=SQR((X1-X0)**2+(Y1-Y0)**2)
220 PRINT "X0= ";X0;" Y0= ";Y0;" R= ";R
225 STOP
230 PRINT "PUNTI ALLINEATI NESSUNA SOLUZIONE
240 DATA 2,-1,0,1,2,3
250 END
```

Figura 3.4

Esempio di esecuzione:

```
DETERMINAZIONE DI UN CERCHIO PASSANTE PER TRE PUNTI

X0= 2    Y0= 1    R= 2
```

Figura 3.5

3.3 PERIMETRO DI UN POLIGONO

Spesso i campi assumono forme geometriche corrispondenti a poligoni qualsiasi (per esempio quadrilateri). Si desidera conoscere la lunghezza del perimetro al fine di calcolare il prezzo di un ipotetico appezzamento.

Problema: Si sono potute rilevare le coordinate cartesiane di ciascun vertice del campo o di una figura simile avente forma poligonale. Costruire un programma che calcola la lunghezza dell'appezzamento.

Soluzione: Il lavoro consiste nel calcolare la lunghezza di ogni lato e nel fare la loro somma. Siano X(I), Y(I) le coordinate del vertice I. Il lato con l'estremo I e I + 1 avrà come lunghezza:

$$\sqrt{(Y(I+1) - Y(I))^2 + (X(I+1) - X(I))^2}.$$

E' sufficiente leggere prima il numero dei vertici (che è uguale al numero dei lati) poi leggere l'insieme X(I), Y(I) e procedere con il calcolo.

Non bisognerà, nel frattempo dimenticare, che l'ultimo lato ha per estremità i vertici N e 1. Da ciò il flow-chart di Fig. 3.6.

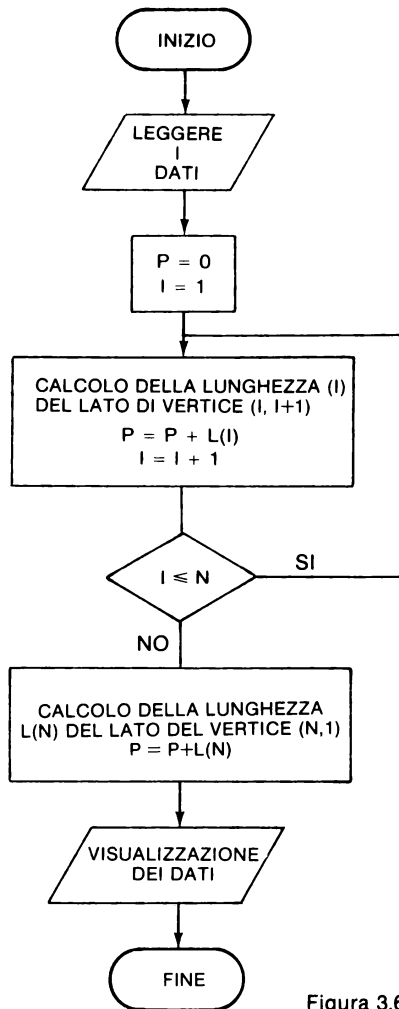



Figura 3.6

Il programma di Fig. 3.7 si divide in più parti.

— un programma principale che non riempie alcuna delle funzioni figuranti nel flow-chart,
— tre sottoprogrammi che effettuano rispettivamente le seguenti funzioni:

  — lettura dei dati,
  — calcolo della lunghezza dei lati del perimetro,
  — stampa dei dati e dei risultati.

Questo procedimento di esecuzione non è giustificato data la semplicità del problema, ma è un valido esempio di come procedere per la soluzione di problemi più complessi.

```
100  REM  CALCOLO DEL PERIMETRO DI UN POLIGONO
110  REM
120  DIM X(100),Y(100),L(100)
130  PRINT "PERIMETRO DI UN POLIGONO"
140  PRINT
150  GOSUB 400
160  GOSUB 500
170  GOSUB 600
180  DATA  5
190  DATA  1,3,4,6,8,6,11,5,11,0
200  STOP
390  REM  LETTURA DEI DATI
400  READ N
410  FOR I = 1 TO N
420  READ X(I),Y(I)
430  NEXT I
440  RETURN
490  REM  CALCOLO DELLA LUNGHEZZA DEL PERIMETRO
500 P = 0
510  FOR I = 1 TO N
520 L(I) =  SQR ((X(I) - X(I + 1)) ^ 2 + (Y(I + 1) - Y(I)) ^ 2)
530 P = P + L(I)
540  NEXT I
550 L(N) =  SQR ((X(N) - X(1)) ^ 2 + (Y(N) - Y(1)) ^ 2)
560 P = P + L(N)
570  RETURN
590  REM  STAMPA DEI RISULTATI
600  PRINT "NUMERO"; TAB( 10);"X"; TAB( 22);"Y"; TAB( 30);"LUNGHEZZA"
610  PRINT
620  FOR I = 1 TO N
630  PRINT I; TAB( 10);X(I); TAB( 22);Y(I); TAB( 30);L(I)
640  NEXT I
650  PRINT
660  PRINT
670  PRINT  TAB( 22);"PERIMETRO = "; TAB( 30);P
680  RETURN
690  END
```

Figura 3.7

```
PERIMETRO DI UN POLIGONO
NUMERO     X                    Y                 LUNGHEZZA
1          1                    3                 4.24264069
2          4                    6                 4
3          8                    6                 3.16227766
4          11                   5                 5
5          11                   0                 10.4403065
                     PERIMETRO =                  37.8452249
BREAK IN 200
]
```

Figura 3.8

3.4 TRACCIAMENTO DI UNA CURVA

L'utilizzo di una stampante o di una macchina da scrivere come strumento per tracciare curve può, a volte, essere utile; in particolare quando non si ha la possibilità di avere un plotter, il problema è quindi trovare e mettere a punto un metodo che permetta di tracciare delle curve in condizioni accettabili.

Problema: Costruire un programma di tracciamento di curve secondo le seguenti istruzioni:

1. Determinare il principio più semplice di lavoro per tracciare una curva $y = f(x)$ per x che varia fra due valori dati A e B.
— Come operare per ridurre l'effetto degli errori di arrotondamento ?

2. Costruire il flow-chart di un sottoprogramma che effettua il tracciamento; e scrivere poi il sottoprogramma stesso:

3. Scrivere in seguito il programma principale che chiama il sottoprogramma e provare a tracciare differenti funzioni come:

— $e^{-1/2x} \cos 2x$ per x che varia da 0 a 10

— $e^{-x}\sqrt{2}$ per x che varia da -2 a + 2.

Soluzione: Tenendo conto del fatto che con una macchina da scrivere è impossibile (salvo casi particolari) effettuare il ritorno della carta, e che è comodo far variare x incrementandolo, il metodo più semplice è quindi quello di scegliere l'asse y orizzontale orientato verso destra e l'asse x verticale orientato verso il basso.

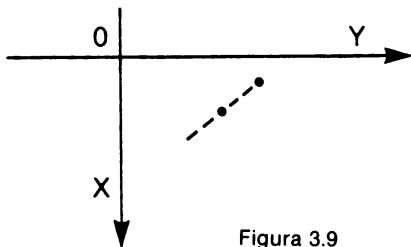



Figura 3.9

Si pongono ora i seguenti problemi:

— posizione degli assi,
— come determinare per il valore dato di y, il numero di spazi da effettuare prima di stampare un punto?

Queste due domande pongono il problema dell'arrotondamento. In effetti, una macchina da scrivere classica può effettuare solo un numero intero di spazi. Per un avanzamento teorico di $y = 8{,}60$ spazi, l'avanzamento reale sarebbe di 9 spazi. Per contro se y ha un valore di 8,40 gli spazi effettuati sarebbero, preferibilmente, 8.

Poichè la funzione TAB normalmente opera per spazi interi, sarà necessario calcolare:

$$Z = \text{INT}(y + 0{,}5) \text{ per ottenere l'arrotondamento}$$

Prima di disegnare il flow-chart, bisogna risolvere il problema della posizione degli assi e del coefficiente di scala: come passare da *y* teorico a *y* reale sul terminale?

Se la scala di variazione di *y* è: $D = Y_{max} - Y_{min}$.

con L che rappresenta il numero massimo di caratteri per linea, si avrà un coefficiente di scala K dato da: K = intero ((L — 1)/D).
Il programma principale potrà quindi comunicare al sottoprogramma:

— Y_{max}, Y_{min}, L,
— i valori di A e B, limiti di variazione di *x*
— l'incremento di H per *x*

Si suppone che Y_{min} sia negativo così che l'asse *x* possa essere stampato nella colonna dalla quale ha origine. I punti della curva saranno rappresentati da punti ortografici.

```
100  REM  PROGRAMMA PER IL TRACCIAMENTO DELLE CURVE
110  REM  LA FUNZIONE FNA RAPPRESENTA LA CURVA DA TRACCIARE
120  DEF  FN A(X) =  EXP ( - X * X * 0.5)
130 A =  - 3
135 L = 40
140 B = 3
150 Y1 = 0
160 Y2 = 1
165 H = 0.2
170  GOSUB 500
180  STOP
500 D = Y2 - Y1
510 K =  INT ((L - 1) / D)
520 Z =  INT (K *  ABS (Y1) + 0.5)
530  FOR X = A TO B STEP H
540 Z1 =  FN A(X) - Y1
550 Z1 =  INT (K * Z1 + 0.5)
560  IF Z1 = Z THEN  PRINT  TAB( Z);"."
570  IF Z1 < Z THEN  PRINT  TAB( Z1);"."; TAB( Z);"I"
580  IF Z1 > Z THEN  PRINT  TAB( Z);"I"; TAB( Z1);"."
590  NEXT X
600  RETURN
```

Figura 3.10

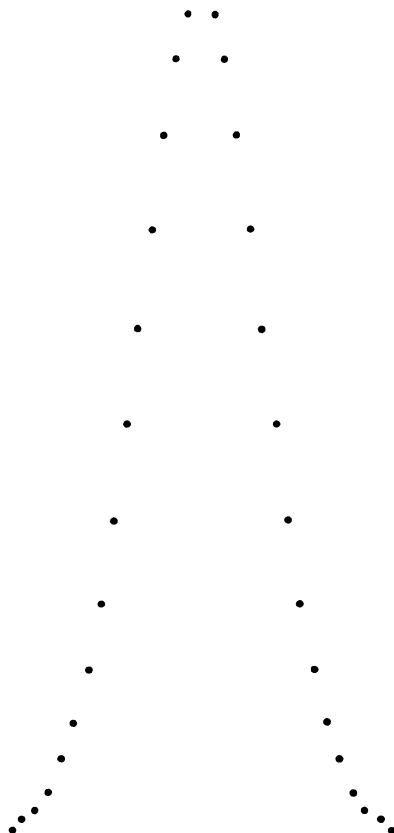



Figura 3.11

3.5 CONCLUSIONI

Questi esercizi danno l'impressione che la programmazione di formule matematiche non presenti difficoltà. In realtà è solo sufficiente scrivere le istruzioni opportune, il flow-chart rimane semplice (flow-chart lineare). L'esercizio di tracciamento di curve suppone già un flow-chart più complesso e una riflessione per la parte "presentazione".

Le suddette impressioni saranno confermate soprattutto quando incontreremo programmi che esigono manipolazioni complesse o che comportano numerosi test.

Questo per esempio è il caso dell'esercizio "regine su una scacchiera" descritto nel Capitolo 10.

CAPITOLO 4

ESERCIZI ORIENTATI ALLA GESTIONE

4.0 INTRODUZIONE

In questo capitolo ci si occuperà di esercizi semplici che presentano un interesse sul piano pratico e sul piano didattico. In effetti per le applicazioni gestionali, si ha frequentemente bisogno di effettuare delle ricerche, fusioni di tabelle o generazione di archivi.

Gli esercizi proposti possono essere ripresi e incorporati in programmi di maggiore complessità al fine di realizzare delle applicazioni complete. Per esempio la sequenza di ricerca data al paragrafo 4.1 può essere utilizzata per generalizzare il programma di funzione in 4.2 e in un certo modo può servire a generalizzare il programma della rubrica telefonica dato in 4.3.

4.1 RICERCA SECONDO IL METODO DI SHELL

Esistono numerosi metodi di ricerca di dati nella memoria del calcolatore. Il metodo più semplice è certamente quello della "bolla" (in Inglese "bubble sort") che si trova in numerosi manuali. Qui viene proposto il metodo di Shell che, senza essere troppo complesso, fornisce le migliori prestazioni per quanto riguarda la velocità di esecuzione riducendo il numero dei confronti necessari.

Si considera una tabella di N numeri da ricercare. Il metodo di Shell consiste nell'effettuare quanto segue:

1. Determinate K in modo che:

$$2^k < n < 2^{k+1}$$

- inizializzare D con il valore $2^k - 1$

2. Effettuare una prima tappa di ricerca facendo variare l'indice I da 1 a N—D

2.1 Provare se $A(I) \leqslant A(I+D)$
se si passare alla tappa successiva (in 3),
se no scambiare A(I) e A(I+D)
Porre k=I e passare alla 2.1

2.2 Provare se $A(K—D) \leqslant A(K)$
se si passare alla tappa successiva (in 3)
se no scambiare, poi porre K = K—D
e passare di nuovo in 2.

3. Continuare i test incrementando I. Quando I ha assunto il valore N e quindi D > 0, ritornare in 2 avendo calcolato a priori D = INT

$$\left(\frac{D-1}{2}\right)$$

Quando D ha assunto il valore 0, la ricerca è terminata.

Problema:

— disegnare il flow-chart di una subroutine di ricerca,
— scrivere un programma che legga una tabella non ordinata e che chiami una subroutine di ricerca.

Soluzione: Il programma può essere scritto facilmente utilizzando la precedente descrizione della tecnica di Shell.

Scambiare Y e Z significa attribuire a Y il valore di Z e a Z il valore di Y. Si scriverà dunque:

```
500 Y = Z
510 Z = Y
```

In questo caso il valore di Y, essendo stato modificato dalla prima istruzione, non darà i risultati sperati nella seconda. Bisognerà inserire prima il contenuto di Y in una variabile ausiliaria X, si scriverà quindi:

```
490 X = Y
500 Y = Z
510 Z = X
```

Questo tipo di scambio si trova nel programma (linee 590-610) e figura anche nel secondo programma della rubrica telefonica (4.5.2).

```
100  DIM A(12)
110 N = 12
120  PRINT "TABELLA INIZIALE"
130  PRINT
140  FOR I = 1 TO N
150  READ A(I)
160  PRINT A(I)"   ";
170  NEXT I
180  GOSUB 500
190  PRINT
195  PRINT
200  PRINT "TABELLA ORDINATA"
210  PRINT
220  FOR I = 1 TO N
230  PRINT A(I)"   ";
240  NEXT I
250  STOP
500 D = 1
510 D = 2 * D
520  IF D < = N THEN 510
530 D =  INT ((D - 1) / 2)
540  IF D = 0 THEN 700
550  FOR I = 1 TO N - D
560 J = I
570 L = J + D
580  IF A(J) <  = A(L) THEN 640
590 X = A(J)
600 A(J) = A(L)
610 A(L) = X
620 J = J - D
630  IF J > 0 THEN 570
640  NEXT I
650  GOTO 530
700  RETURN
800  DATA  3,-1,4,10,8,9,5,-10,-5,25,22,7
900  END
```

Figura 4.2

Esempio di esecuzione:

```
TABELLA INIZIALE

3  -1  4  10  8  9  5  -10  -5  25  22  7

TABELLA ORDINATA

-10  -5  -1  3  4  5  7  8  9  10  22  25
```

Figura 4.3

4.2 FUSIONE DI DUE TABELLE

Si richiede la fusione di due tabelle A e B contenenti numeri in ordine crescente, in una terza tabella C anch'essa ordinata secondo lo stesso criterio.

Esempio:	A	3	4	6	18			
	B	—1	0	5				
Per ottenere:								
	C	—1	0	3	4	5	6	18

Metodo da utilizzare: Utilizzare tre indici I, J, K per ciascuna tabella, questi indici saranno inizialmente "settati" a 1.

Se $a_i \leqslant b_j$ inserire a_i in C_k,
incrementare i e k.

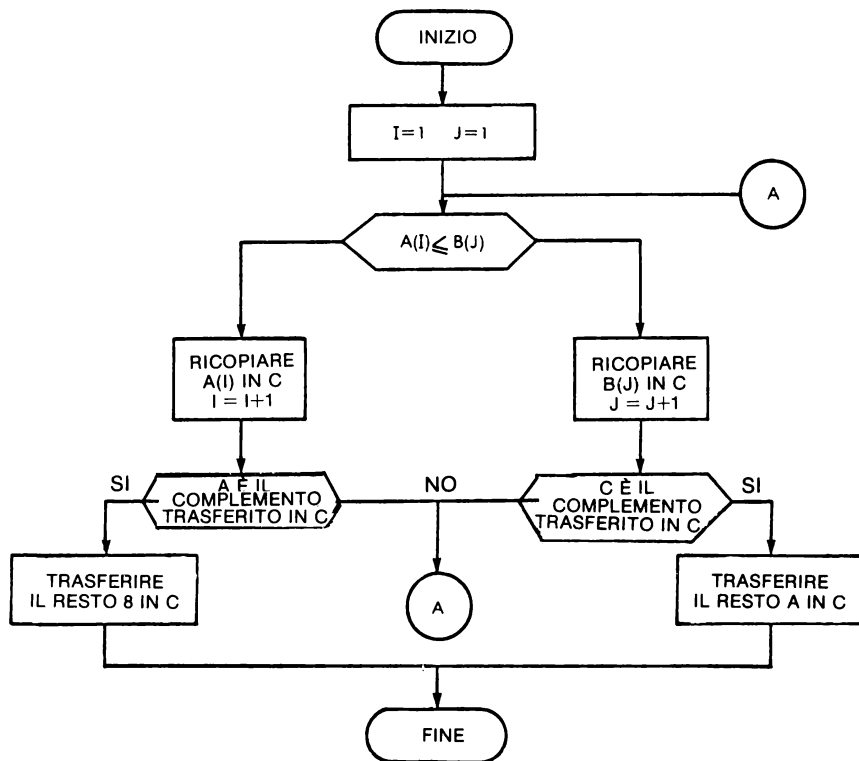
Se $a_i > b_j$ inserire b_j in C_k,
incrementare j e k,

quando una delle tabelle A e B è stata completamente trasferita in C, bisogna trasferire il resto dell'altra tabella in C.

Problema: Disegnare il flow-chart,
scrivere una subroutine in BASIC.

Domande: *a*) Come fare se le tabelle A e B non sono ordinate?
b) Come si può adattare il programma alla fusione di due files sequenziali ordinati?

Figura 4.4

Soluzione: Il metodo proposto porta al flow-chart 4.4 che necessita di qualche spiegazione prima di poter cominciare la scrittura del programma. Il passaggio al flow-chart di Fig.4.5 è molto semplice a condizione che non si dimentichi di utilizzare tre indici.

I indice per A,
J indice per B,
K indice per C.

Nel programmare per evitare un GO TO, è preferibile inizialmente porre K a 0 e l'istruzione K = K + 1 situata nel mezzo del flow-chart all'inizio dell'algoritmo.

Così anche i due piccoli loop possono essere scritti per mezzo delle istruzioni FOR ... NEXT a condizione di utilizzare una variabile ausiliaria.

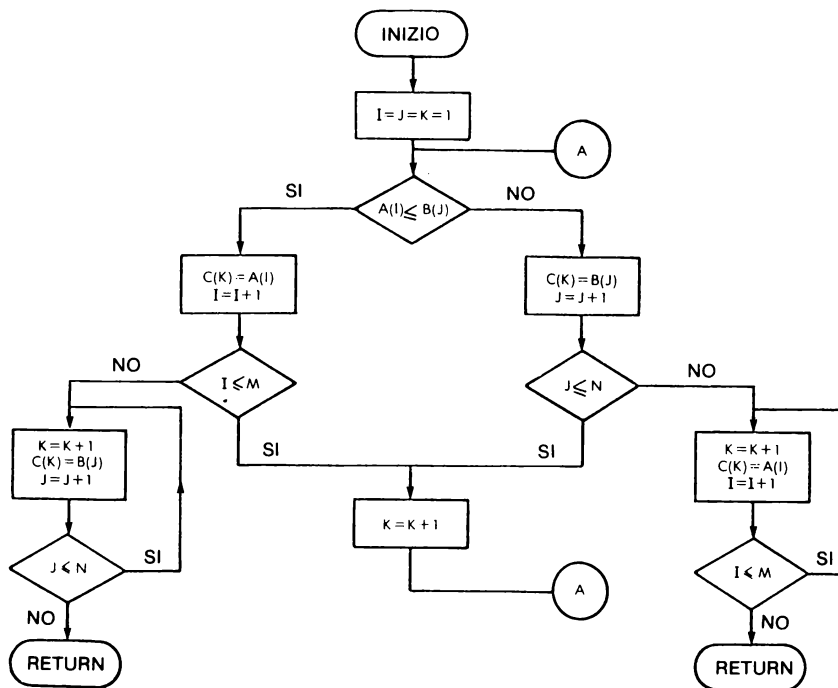
Figura 4.5

```
100  DIM A(100),B(100),C(100)
110  REM  LETTURA TABELLA A
120  READ M
130  PRINT "TABELLA A: "
140  FOR I = 1 TO M
150  READ A(I): PRINT A(I);: NEXT I
160  PRINT
170  PRINT
180  REM  LETTURA TABELLA B
190  PRINT "TABELLA B: "
200  READ N
210  FOR I = 1 TO N
220  READ B(I): PRINT B(I);: NEXT I
230  PRINT
240  PRINT
250  GOSUB 300
260  PRINT "TABELLA C: "
270  FOR I = 1 TO M + N
280  PRINT C(I);: NEXT I
290  STOP
295  REM  SUBROUTINE PER LA FUSIONE DI Z E B
300 I = 1:J = 1:K = 1
310  IF A(I) >  = B(J) THEN 350
320 C(K) = A(I):I = I + 1
330  IF I > M THEN 390
340 K = K + 1: GOTO 310
350 C(K) = B(J):J = J + 1
360  IF J <  = N THEN 340
365  REM  COPIA IL RESTO DI A IN C
370 K = K + 1:C(K) = A(I):I = I + 1
380  IF I <  = M THEN 370ELSE RETURN
385  REM  COPIA IL RESTO DI B IN C
390 K = K + 1:C(K) = B(J):J = J + 1
400  IF J <  = N THEN 390ELSE RETURN
410  DATA  5
420  DATA  4,7,9,12,45
430  DATA  4
440  DATA  -1,5,6,60
450  END
]
```

Figura 4.6

Esempio di esecuzione:

```
TABELLA A:
 4 7 9 12 45
TABELLA B:
 -1 5 6 60
TABELLA C:
 -1 4 5 6 7 9 12 45 60
BREAK IN 290
```

Figura 4.7

ESTENSIONI DI QUESTO PROGRAMMA

Prima estensione: Come adattarlo a due tabelle A e B non ordinate. La prima idea che consiste nel fare una tabella unica non ordinata e di effettuare in seguito un ordinamento. Questo metodo conduce a un ordinamento più lungo di quello del secondo metodo che consiste nell'ordinare prima A e B ed effettuare poi la fusione (1).

INIZIO
APERTURA FILE
LEGGERE ARTICOLO FILE A
SI
A
FINE FILE A
NO
LEGGERE ARTICOLO FILE B
SI
B
FINE FILE B
NO
SI
CODICE A E CODICE B
NO
SCRIVERE ARTI-COLO B IN C
SCRIVERE ARTI-COLO A IN C
A
B
RICOPIARE TESTO DI B IN C
RICOPIARE TESTO DI A IN C
CHIUSURA FILE
FINE

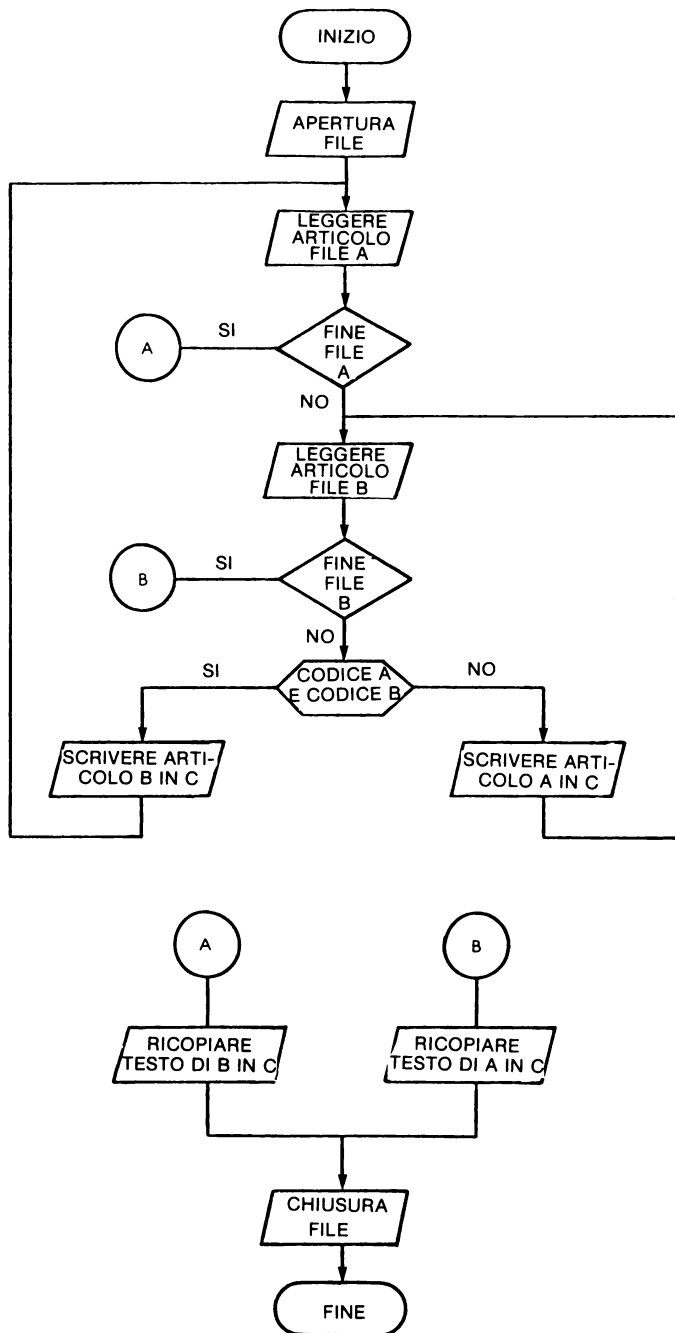



Figura 4.8

Per realizzare questi ordinamenti preliminari, si potrà utilizzare la sequenza precedente (linee dalla 500 alla 700 che bisognerà ripetere una volta per la tabella A e una volta per la tabella B) poichè la maggior parte degli interpreti e dei compilatori BASIC non accettano i sottoprogrammi con trasmissione di parametri (2).

Seconda estensione: Fusione di due files sequenziali.

Il flow-chart di Fig.4.4 costituisce un eccellente punto di partenza. Bisognerà includere le istruzioni di lettura e scrittura, pensare che raramente si conosce a priori il numero "degli articoli" dei files, dunque si prevederanno test periodici per la conclusione dei files.

Questa estensione conduce al flow-chart di Fig.4.8. La programmazione dipende dal sistema utilizzato. In effetti le istruzioni per i files non sono normalizzate in BASIC.

4.3 GIORNO DELLA SETTIMANA

Partendo dalla data: GIORNO, MESE, ANNO, indicare di quale giorno della settimana si tratta. Questo problema è stato risolto con numerosi metodi, noi ve ne proponiamo uno:

— Calcolare il valore del termine di correzione N, normalmente N = 0 ma se il mese è Gennaio o Febbraio N avrà per valore:

2 se l'anno non è bisestile
1 se l'anno è bisestile

— Calcolare in seguito il "codice giorno" con:

C: = intero (365,25 x A2) + intero (30,56 x M) + J + N,

A1 rappresenta le due prime cifre dell'anno e,
A2 le ultime due cifre,
per esempio 1979 dà A1=19 e A2=79

— Calcolare poi il numero S del giorno della settimana con

$$S = C + 3 - 7 x \text{ intero } \left(\frac{c+2}{7}\right)$$

corrisponde a Lunedì,
S = 2 corrisponde a Martedì,
etc.
S = 7 corrisponde a Domenica.

---

(1) Per maggiori dettagli consultare pubblicazioni specializzate in algoritmi di ordinamento.

(2) Questo costituisce un notevole vantaggio del FORTRAN sul BASIC.

Un anno è bisestile se:

A2 ≠ 0 e A2 divisibile per 4, oppure A2 = 0 e A1 divisibile per 4.

Esempio: 1900 non è bisestile poichè 19 non è divisibile per 4.
1984 è bisestile poichè 84 è divisibile per 4.

4.3.1 Problema: Scrivere un programma che a partire dalla data G, M, A stampi il giorno della settimana corrispondente.

4.3.2 Soluzione: Per poter utilizzare il programma in modo semplice, si costruirà un programma che domandi una nuova data fino a che il giorno introdotto sia negativo o nullo (0). Procedendo dal metodo proposto, la costruzione del flow-chart non presenterà difficoltà.

E' necessario conoscere:
— come calcolare A1 e A2,
— se l'anno è bisestile.

Si noti che A1 è uguale al quoziente della "divisione intera di A : 100" quindi:

A1= INT (A/100)

A2 rappresenta il resto della divisione intera precedente, quindi:

A2 = A — 100 x A1

Per sapere se A2 è divisibile per 4, è sufficiente calcolare il resto R per:

R = A2 — 4 x INT (A2 / 4)

Questo tipo di istruzione appare sovente negli esercizi del Capitolo 2. Si può ora scrivere una parte importante del programma fino al calcolo di S incluso. Bisogna studiare come si realizzerà la visualizzazione dei dati, vengono presentati differenti casi:

Primo caso: Il sistema accetta delle tabelle di stringhe di caratteri. In questo caso è sufficiente utilizzare le seguenti istruzioni:

```
DIM G$ (7)
G$ (1) = "LUNEDI"
G$ = "DOMENICA"
```

e stampare il giorno con:

```
PRINT G$ (S)
```

INIZIO
LETTURA DEI DATI
SI
J≤ 0
FINE
NO
CALCOLO DI A1 E A2
N = 0
NO
M > 2
SI
N=2
SI
ANNI BISESTILI?
NO
N=1
CALCOLO DI C
CALCOLO DI S
VISUALIZZAZIONE GIORNO DELLA SETTIMANA

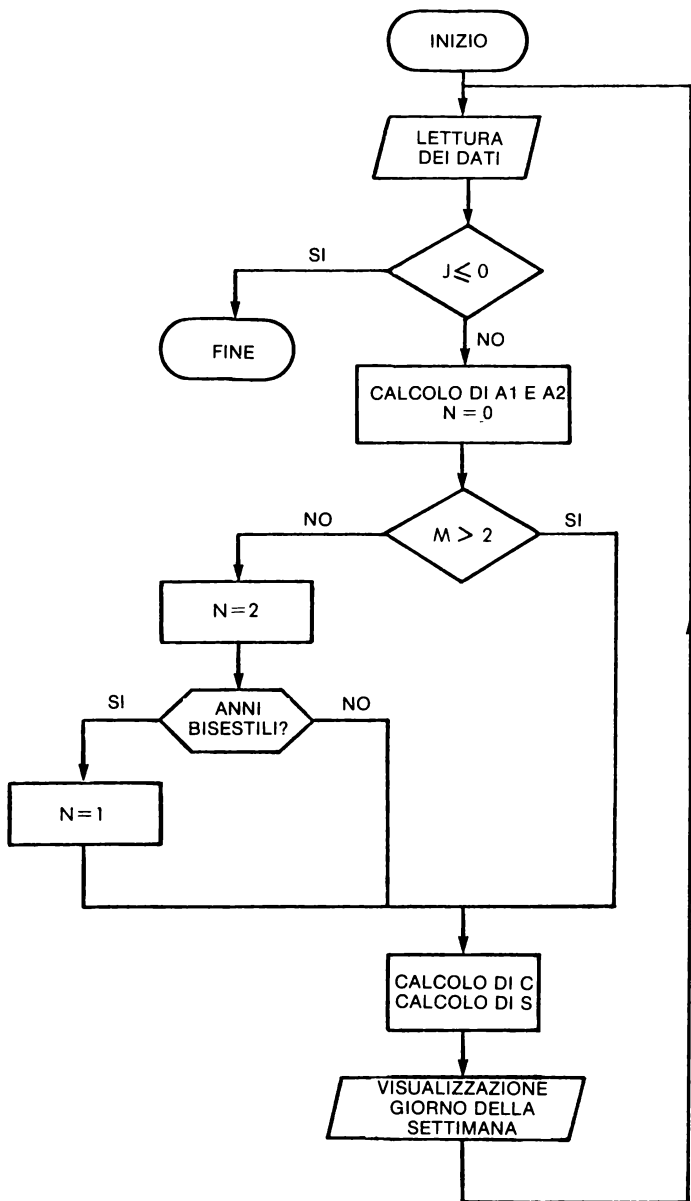



Figura 4.9

Secondo caso: Il sistema non accetta delle tabelle di stringhe. Si potrà allora impostare una stringa di caratteri abbastanza lunga per contenere tutti i giorni della settimana.

Il giorno più lungo è MERCOLEDI' con 9 caratteri. Una stringa di 9 x 7 = 63 caratteri potrà dunque essere inizializzata convenientemente è sarà sufficiente per stampare la sottostringa per mezzo di un'istruzione del tipo:

PRINT SUBSTR (G$,8 * S — 8,9)

oppure

PRINT MID $ (G$,8 * S — 8,9)

```
90  DIM G$(56)
95 G$ = "LUNEDI   MARTEDI   MERCOLEDIGIOVEDI   VENERDISABATO  DOMENICA"
100  REM  CALCOLO DEL GIORNO DELLA SETTIMANA
110  REM  S RAPPRESENTA IL NUMERO DEL GIORNO (1 PER LUNEDI E 7 PER
115  REM  DOMENICA)
120  INPUT "IMPOSTARE LA DATA ";G,M,A
130 A1 =  INT (A / 100)
140 A2 = A - 100 * A1
150 N = 0
160  IF M > 2 THEN 300
165 N = 2
170  IF A2 = 0 THEN 220
180 R = A2 - 4 *  INT (A2 / 4)
190  IF R <  > 0 THEN 300
200 N = 1
210  GOTO 300
220 R = A1 - 4 *  INT (A1 / 4)
230  IF R = 0 THEN N = 1
300 C =  INT (365.25 * A2) +  INT (30.56 * M) + N + G
310 S = 3 + C - 7 *  INT ((C + 2) / 7)
320  PRINT  MID$ (G$,9 * M - 8,9)
330  PRINT
340  GOTO 120
```

Figura 4.10

Nota: Nel corso dell'esercizio 4.4 vedremo l'utilizzazione di una funzione utente per ridurre il numero di istruzioni del programma.

Il MICROSOFT BASIC non utilizza la funzione SUBSTR ma la funzione MID$.

```
IMPOSTARE LA DATA ?12,4,1979
GIOVEDI
IMPOSTARE LA DATA ?13,4,1979
VENERDI
IMPOSTARE LA DATA ?15,8,1979
MERCOLEDI
IMPOSTARE LA DATA ?
BREAK IN 200
```

Figura 4.11

4.3.3 Analisi di un programma

Il programma di Fig.4.12 permette ugualmente di ottenere il giorno della settimana utilizzando un altro metodo.

```
100  REM  PROGRAMMA PER CALCOLARE IL GIORNO DELLA SETTIMANA
110  DIM G$(7)
120 G$(1) = "LUNEDI"
130 G$(2) = "MARTEDI"
140 G$(3) = "MERCOLEDI"
150 G$(4) = "GIOVEDI"
160 G$(5) = "VENERDI"
170 G$(6) = "SABATO"
180 G$(7) = "DOMENICA"
190  PRINT "IMPOSTARE LA DATA ";
200  INPUT G,M,A
210  GOSUB 500
220  PRINT G$(Z)
230  GOTO 190
500  IF A <  = 1752 THEN 620
510 N =  INT (0.6 + 1 / M)
520 L = A - N
530 P = M + 12 * N
540 D = L / 100
550 A1 =  INT (D)
560 Z1 =  INT (D / 4)
570 Z3 =  INT (5 * L / 4)
580 Z4 =  INT (13 * (P + 1) / 5)
590 Z = Z4 + Z3 - A1 + Z1 + G + 5
600 Z = Z - (7 *  INT (Z / 7)) + 1
610  RETURN
620  PRINT "L'ANNO DEV'ESSERE POSTERIORE A 1752 "
630  STOP
640  END
```

Figura 4.12

```
IMPOSTARE LA DATA 5,2,1979
LUNEDI'
IMPOSTARE LA DATA 13,5,1979
DOMENICA
IMPOSTARE LA DATA 14,7,1979
SABATO
IMPOSTARE LA DATA
```

Figura 4.13

Domande:

1. Cosa significa l'istruzione della linea 510 ?
2. Come viene interpretata l'istruzione di linea 530 ?
3. E' possibile, senza modificare il metodo e senza aumentare il numero delle operazioni, ridurre il numero delle istruzioni del programma ?

Soluzione:

1. Istruzione di linea 510

— se M vale 1 o 2, N assume il valore intero (0.6 + 1) o intero (0.6 + 1/2), che in ogni caso dà 1,
— se M vale 3, 4, etc., N assume il valore 0.

Questo è dunque da rapportare al programma precedente dove si teneva conto delle conseguenze provocate dall'anno bisestile.

2. Istruzione della linea 530

— P corrisponde al numero del mese se questo è Marzo, Aprile, etc. fino a Dicembre. Per Gennaio e Febbraio P assumerà rispetivamente il valore 13 e 14.
D-A1 nell'istruzione della linea 590 ha un valore V che è V=0 se si trova in un anno secolare.

$0 < V < 1$ se si trova in un altro anno.

3. Poichè le variabili A1, Z1, Z3 e Z4 sono utilizzate solo nel calcolo di Z nella linea 590, si possono sopprimere le istruzioni 560, 570 e 580 e scrivere in 590:

590 Z = INT (13 · (P + 1) / 5) + INT (S · L / 4) — INT (0) + INT (D / 4) + J + 5

4.4 DIFFERENZA TRA DUE DATE

Per conoscere la differenza tra due date si può calcolare "il codice giorno" di ogni data e fare la differenza. Si ottiene così il numero dei giorni che le separano. Questa informazione è indispensabile quando si debba calcolare l'ammontare di interessi per un periodo qualunque.

Problema: Riferendosi al programma precedente costruire un nuovo programma che calcoli la differenza tra due date.

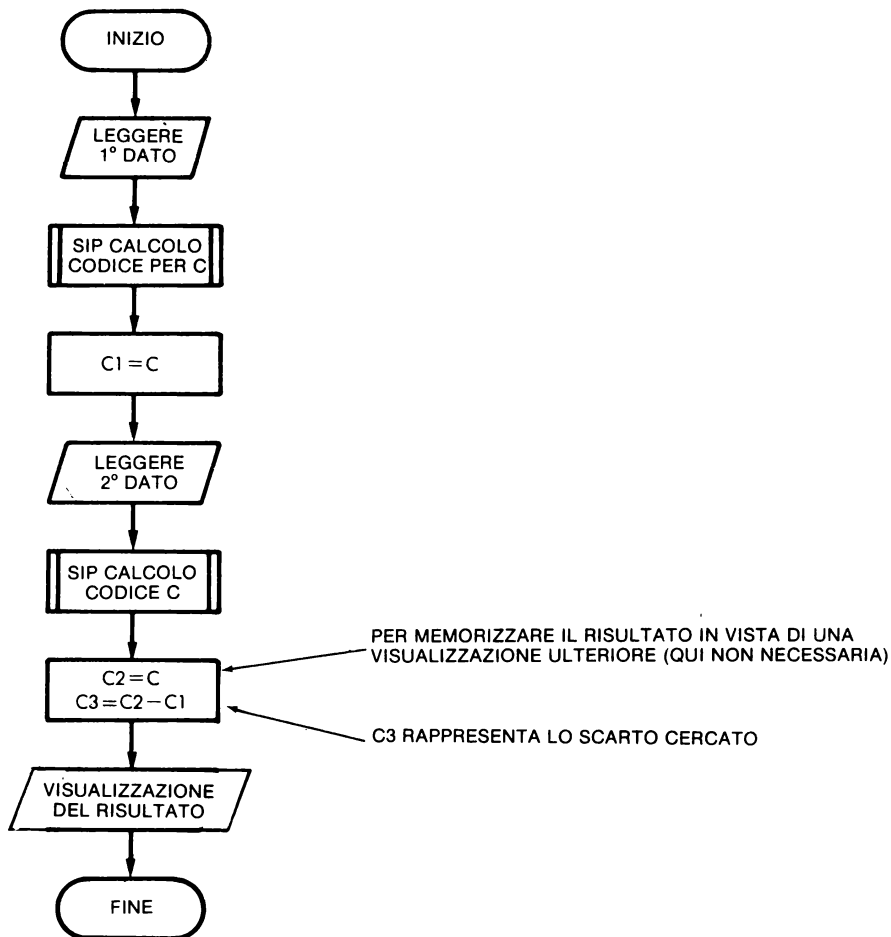



Figura 4.14

Soluzione: E' possibile verificare l'importanza del sottoprogramma, si dovrà in effetti calcolare per due volte successive il codice giorno e sarà preferibile costruire un sottoprogramma per questo calcolo. Si può dunque realizzare il flow-chart di Fig.4.14.

Riprendendo la parte del programma di Fig.4.10 che calcola il "codice giorno", si costruisce rapidamente il programma 4.15.

4.5 RUBRICA TELEFONICA

Il BASIC presenta alcuni vantaggi in confronto ad altri linguaggi, come l'"anti-

```
100  REM  CALCOLO DELLA DIFFERENZA TRA DUE DATE
105  REM  AUTORE  J.P. LAMOITIER
110  DEF  FN R(A,D) = A - D * INT (A / D)
120  INPUT "IMPOSTARE LA PRIMA DATA ";G,M,A
130  GOSUB 500
140 C1 = C
150  INPUT "IMPOSTARE LA SECONDA DATA  ";G,M,A
160  GOSUB 500
170  PRINT
180  PRINT "TRA LE DUE DATE VÍ SONO ";C - C1;" GIORNI"
200  STOP
500 A2 =  FN R(A,100)
510 N = 0
520  IF M > 2 THEN  GOTO 570
525  IF M <  = 2 THEN N = 2
530  IF A2 = 0 THEN  GOTO 550
535  IF A2 <  > 0 THEN R =  FN R(A2,4)
540  IF R <  > 0 THEN  GOTO 570
545  IF R = 0 THEN  GOTO 560
550  IF  FN R(A1,4) <  > 0 THEN 570
560 N = 1
570 C =  INT (365.25 * A2) +  INT (30.56 * M) + N + G
580  RETURN
590  END
```

Figura 4.15

Esempio d'esecuzione:

```
IMPOSTARE LA PRIMA DATA 23,2,1979
IMPOSTARE LA SECONDA DATA 30,6,1980
TRA LE DUE DATE VI SONO 493 GIORNI
```

Figura 4.16

co" FORTRAN, tra gli altri quello di trattare molto semplicemente le stringhe di caratteri. I due esercizi seguenti mostrano come si possono facilmente manipolare le stringhe di caratteri in relazione ad applicazioni pratiche.

4.5.1 Esercizio 1: Scrivere un programma che legga delle linee DATA contenenti cognome, nome, codice, numero telefonico e che stampi in modo preciso dette linee. Si daranno a queste stringhe di caratteri rispettivamente i nomi N$, P$, B$ e T. Per questo esercizio si supporrà che la lista dei dati sia in ordine alfabetico.

Soluzione: Il flow-chart di principio è semplice poichè è sufficiente leggere e stampare senza effettuare elaborazioni. La sola difficoltà è conoscere quando la lista dei dati è terminata. Due metodi possono essere convenientemente utilizzati:

— utilizzare un'ultima registrazione particolare che "precede" la fine della lista,

— utilizzare l'istruzione IF...END propria di alcuni BASIC.

Qui verrà utilizzato il primo metodo che è alla portata di tutti i sistemi.

Alla fine della lista, si aggiunge un nome particolare che è "ZZZ" e quindi facilmente riscontrabile. In questo modo è possibile conoscere il termine della lista.

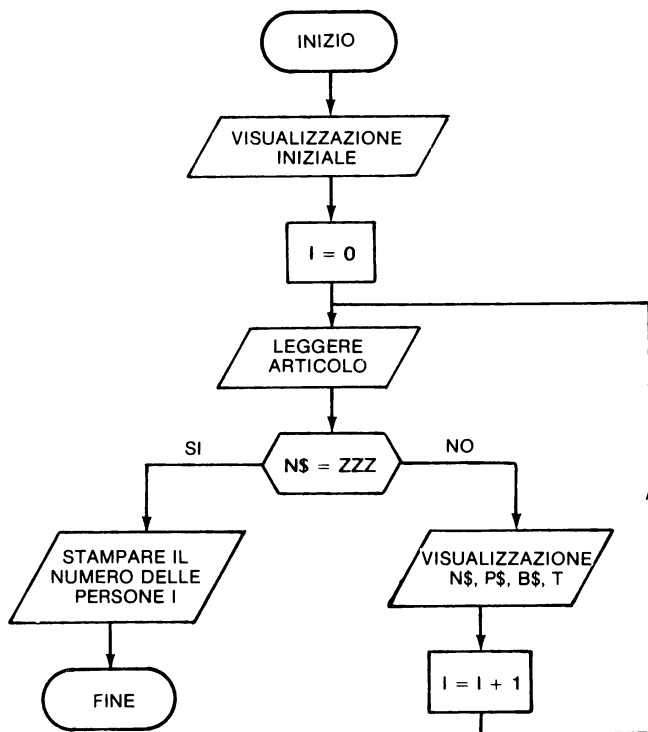



Figura 4.17

Tutto questo conduce al flow-chart di Fig.4.17.

In questo flow-chart è stata aggiunta una variabile I che "conta" il numero reale di persone contenute nella lista. Così, per liste importanti, contando il numero delle linee di uscita (il numero delle persone) è possibile prevedere dei salti di pagina al fine di ottenere una buona presentazione.

Questo programma, che non effettua alcuna ricerca, permette tuttavia di ottenere un elenco ordinato se i nomi sono stati introdotti correttamente, per questo motivo le linee sono numerate di 10 in 10 così che altre linee possano essere inserite dove necessario.

Mediante qualche piccola modifica (a livello di istruzione di lettura) si può lavorare come per un file sequenziale, che non limita la dimensione dell'elenco. Bisognerà nel frattempo assicurarsi del continuo aggiornamento del file.

Nota: Alcuni BASIC permettono di "testare" la fine del file con l'istruzione IF....END; ciò evita di dover impostare una registrazione fittizia (come ZZZ). Il flow-chart è presente in Fig.4.19.

```
100  REM  PROGRAMMA PER LA RUBRICA TELEFONICA
110  REM
120  REM
130  REM
140  REM
150  PRINT  TAB( 6),"RUBRICA TELEFONICA"
160  PRINT
170  PRINT "COGNOME"; SPC( 6);"NOME"; SPC( 7);"NUMERO"; SPC( 9);"INTERNO"
180  PRINT
190 I = 0
200  READ C$,N$,B$,T
210  IF C$ = "ZZZ" THEN 250
220  PRINT C$; SPC( 6);N$; SPC( 10);B$; SPC( 14);T
230 I = I + 1
240  GOTO 200
250  PRINT
260  PRINT "NUMERO DELLE PERSONE = ";I
270  DATA  ROSSI,ANDREA,"3",310
280  DATA  ROSSI,GIANNI,"3",340
290  DATA  VERDI,LUIGI,"5",400
300  DATA  VINCI,CLAUDIO,"4",360
900  DATA  ZZZ,Z,3,4
```

```
                  RUBRICA TELEFONICA

COGNOME      NOME       NUMERO         INTERNO

ROSSI        ANDREA        3              310
ROSSI        GIANNI        3              340
VERDI        LUIGI        5              400
VINCI        CLAUDIO        4               360

NUMERO DELLE PERSONE = 4
```

Figura 4.18

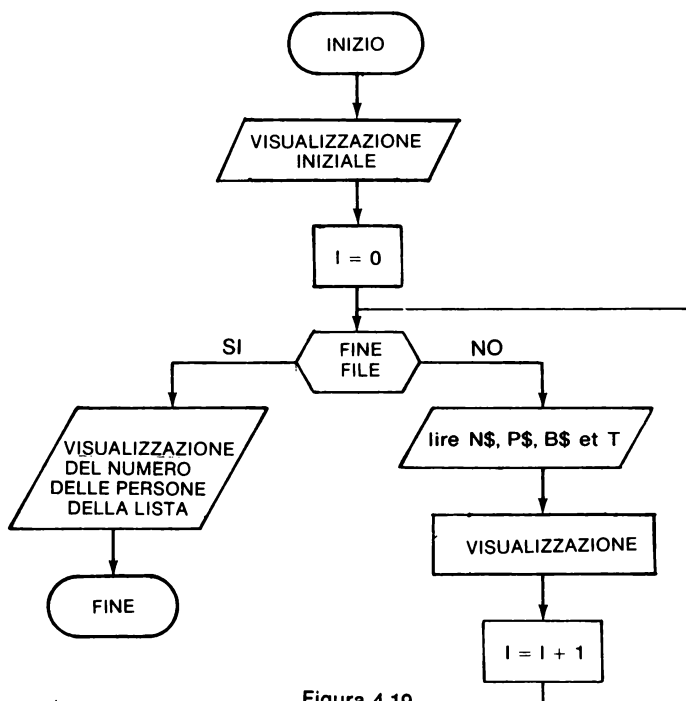



Figura 4.19

4.5.2 Esercizio 2: Si tratta ora di costruire un programma più complesso che propone un "menu" all'utilizzatore:

- ricerca di un cognome,
- ricerca di un cognome e nome,
- ricerca del solo nome,
- ricerca del numero telefonico,
- estrazione delle persone che hanno un interno,

e che effettui in seguito il lavoro richiesto dall'utilizzatore. Per la ricerca si può riprendere la sequenza di ricerca già esaminata, a condizione che venga adattata al problema da risolvere.

Problema: Costruire un programma che legga i dati che si suppone siano immagazzinati nelle istruzioni DATA del programma, e che proponga poi il seguente menu:

In funzione della risposta data dall'utilizzatore, il programma effettua il lavoro selezionato poi propone di nuovo il menu. Il programma termina se l'utilizzatore introduce un numero $\leqslant 0$ o $\geqslant 6$.

Nota: Il BASIC non accetta, in generale, delle subroutine con trasmissione di parametri, questo è invece possibile con numerosi linguaggi come ALGOL, FORTRAN, etc. Si incontreranno difficoltà nèl costruire la subroutine di ricerca che possa funzionare correttamente in tutti i casi. Si potrà quindi considerare di trasferire "la chiave di ricerca" in una tabella ausiliaria. Bisognerà far attenzione a non dimenticare le permutazioni.

Soluzione: Questo problema non presenta difficoltà particolari a condizione di lavorare con metodo. Si costruirà prima un flow-chart generale non dettagliato (Fig.4.20).

Per utilizzare il sottoprogramma di ricerca già esaminato, sarà necessario ogni volta preparare i dati che possono essere usati dallo stesso. E' stata scelta la seguente convenzione:

I dati sono C$ (I) cognome
N$ (I) nome
D$ (I) codice
T$ (I) numero telefonico

Una tabella ausiliaria B$ (I) riceve prima gli elementi da ricercare e in seguito viene effettuata la ricerca. Prima di effettuare la ricerca sul cognome si esegue la sequenza:

```
FOR I = 1 TO N
    B$ (I) = C$ (I)
    NEXT I
```

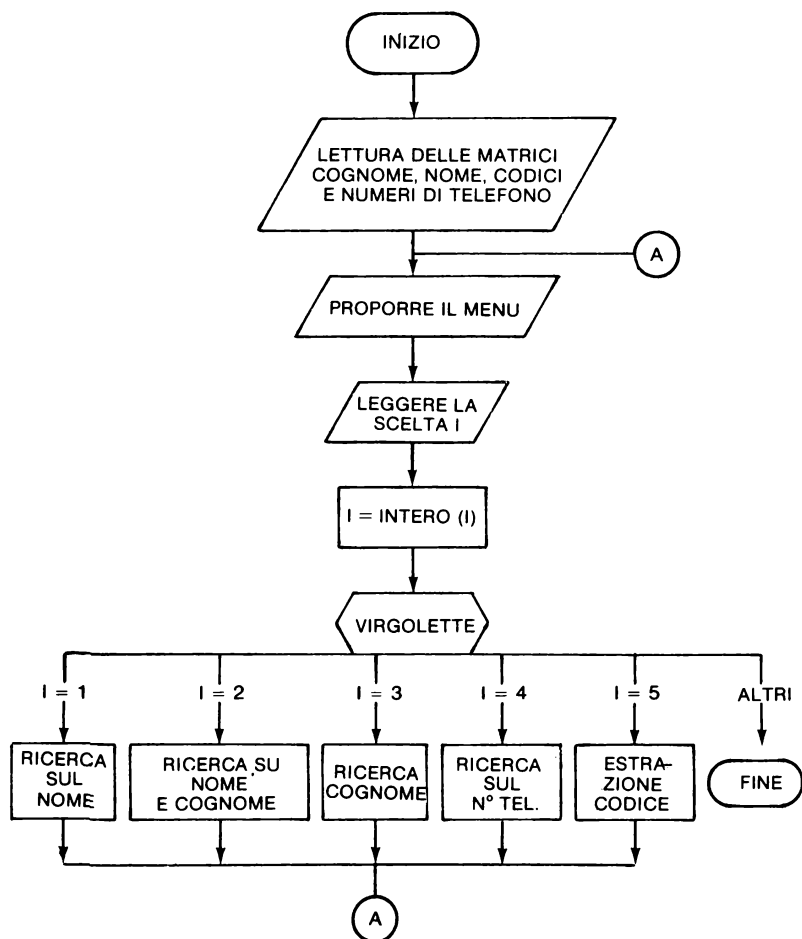



Figura 4.20

Questo presuppone dunque che si debba effettuare una ricerca alfabetica, il che non è una difficoltà in BASIC con i codici ASCII o EBCDIC.

Al contrario per la ricerca del cognome e del nome, sarà necessario porre in B$ la concatenazione C$ + N$, ma in questo modo si corrono dei rischi; si considerino i seguenti nomi:

ROSSI UGO
ROSSINI ALDO

per concatenazione semplice si avrà:

ROSSIUGO
ROSSINIALDO

e il confronto darà ROSSINIALDO < ROSSIUGO.

Per evitare tale rischio si inserirà tra cognome e nome un carattere "spazio" che in ASCII possiede un valore inferiore alla lettera A. Si avrà quindi:

ROSSI UGO
ROSSINI ALDO

così il confronto fornirà i risultati voluti. Per inserire il carattere "spazio" è sufficiente scrivere:

470 B$ (I) = C$ (I) + "∧" + N$ (I)

↑ spazio tra le virgolette

```
100  REM  PROGRAMMA DI RUBRICA TELEFONICA
110  REM
120  DIM S$(100),C$(100),N$(100),T(100),B$(100)
130  GOSUB 800
140  PRINT "INDICARE IL TIPO DI RICERCA"
150  PRINT "1 = RICERCA DEL COGNOME"
160  PRINT "2 = RICERCA DEL COGNOME E NOME"
170  PRINT "3 = RICERCA DEL NOME"
180  PRINT "4 = RICERCA DEL NUMERO TELEFONICO"
190  PRINT "5 = ESTRAZIONE DEL NUMERO DI CODICE"
200  INPUT I
210  REM  PROVA DI VALIDITA'
220 I =  INT (I)
230  IF I <  = 0 THEN  STOP
240  IF I >  = 6 THEN  STOP
250  REM  SELEZIONE DEL LAVORO
260  ON I GOTO 400,460,520,580,640
270  STOP
```

```
390  REM  RICERCA DEL COGNOME
400  FOR I = 1 TO N
410 B$(I) = C$(I)
420  NEXT I
430  GOSUB 1000
440  GOTO 140
```

si domanda solo la ricerca sul cognome

```
450  REM  RICERCA DEL COGNOME E DEL NOME
460  FOR I = 1 TO N
470 B$(I) = C$(I) + " " + N$(I)
480  NEXT I
490  GOSUB 1000
500  GOTO 140
```

si domanda solo la ricerca sul cognome e il nome

```
510  REM  RICERCA DEL NOME
520  FOR I = 1 TO N
530 B$(I) = N$(I)
540  NEXT I
550  GOSUB 1000
560  GOSUB 140
```

si domanda la ricerca sul nome

```
570  REM  RICERCA DEL NUMERO TELEFONICO
580  FOR I = 1 TO N
590 B$(I) =  STR$ (T(I))
600  NEXT I
610  GOSUB 1000
```

ricerca sul numero telefonico

```
620  GOTO 140
630  REM  ESTRAZIONE DEL NUMERO DI CODICE
640  PRINT "IMPOSTARE IL CODICE SCELTO ";
650  INPUT A$
660  PRINT
670 J = 0
680  PRINT "ELENCO DEI NOMI CON CODICE ";A$
690  PRINT
700  FOR I = 1 TO N
710  IF S$(I) <  > A$ THEN 720
715  PRINT C$(I),N$(I),S$(I),T(I)
717 J = J + 1
720  NEXT I
730  PRINT
740  PRINT "IN TOTALE ";J;" NOMI NELL'ELENCO"
750  GOTO 140
790  REM  SOTTOPROGRAMMA DI LETTURA
800  READ N
810  FOR I = 1 TO N
820  READ C$(I),N$(I),S$(I),T(I)
830  IF C$(I) = "ZZZ" THEN 860
840  NEXT I
860 N = I - 1
870  RETURN
990  REM  SOTTOPROGRAMMA DI RICERCA SECONDO IL METODO DI SHELL
1000 D = 1
1010 D = 2 * D
1020  IF D <  = N THEN 1010
1030 D =  INT ((D - 1) / 2)
1040  IF D = 0 THEN 1400
1050  FOR I = 1 TO N - D
1060  FOR J = I TO 1 STEP  - D
1070 L = J + D
1080  IF B$(J) <  = B$(L) THEN 1350
1090 X$ = C$(J)
1100 C$(J) = C$(L)
1110 C$(L) = X$
1120 X$ = N$(J)
1130 N$(J) = N$(L)
1140 N$(L) = X$
1150 X$ = S$(J)
1160 S$(J) = S$(L)
1170 S$(L) = X$
1180 X = T(J)
1190 T(J) = T(L)
1200 T(L) = X
1205 X$ = B$(J)
1206 B$(J) = B$(L)
1207 B$(L) = X$
1210  NEXT J
1350  NEXT I
1360  GOTO 1030
1400  GOSUB 1500
1410  RETURN
1490  REM  SEQUENZA DI STAMPA
1500  PRINT
1510  PRINT "COGNOME","NOME","CODICE","NUMERO"
1520  PRINT
1530  FOR K = 1 TO N
1540  PRINT C$(K),N$(K),S$(K),T(K)
1550  NEXT K
1560  PRINT
1570  PRINT "IN TOTALE ";N;" NOMI NELL'ELENCO"
1580  RETURN
2000  DATA  10
2010  DATA  ROSSI,PIERO,BE,100
2020  DATA  ROSSINI,GIANNI,BE,110
2030  DATA  VERDI,CLAUDIO,LABO,310
2040  DATA  ZANI,DARIO,FA,115
2050  DATA  ZINNI,FRANCO,COM,300
2060  DATA  CARLI,PAOLO,SEC,301
2070  DATA  ZZZ,Z,Z,0
```

estrazione del numero di interno

Figura 4.21

Esempio di esecuzione:

```
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?1



COGNOME           NOME              CODICE            NUMERO

CARLI             PAOLO             SEC               301
ROSSI             PIERO             BE                100
ROSSINI           GIANNI            BE                110
VERDI             CLAUDIO           LABO              310
ZANI              DARIO             FA                115
ZINNI             FRANCO            COM               300

IN TOTALE 6 NOMI NELL'ELENCO
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?2



COGNOME           NOME              CODICE            NUMERO

CARLI             PAOLO             SEC               301
ROSSI             PIERO             BE                100
ROSSINI           GIANNI            BE                110
VERDI             CLAUDIO           LABO              310
ZANI              DARIO             FA                115
ZINNI             FRANCO            COM               300

IN TOTALE 6 NOMI NELL'ELENCO
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?3



COGNOME           NOME              CODICE            NUMERO

VERDI             CLAUDIO           LABO              310
ZANI              DARIO             FA                115
ZINNI             FRANCO            COM               300
ROSSINI           GIANNI            BE                110
CARLI             PAOLO             SEC               301
ROSSI             PIERO             BE                100
```

Figura 4-22 (continua)

```
IN TOTALE 6 NOMI NELL'ELENCO
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?4
COGNOME              NOME                 CODICE               NUMERO

ROSSI                PIERO                BE                   100
ROSSINI              GIANNI               BE                   110
ZANI                 DARIO                FA                   115
ZINNI                FRANCO               COM                  300
CARLI                PAOLO                SEC                  301
VERDI                CLAUDIO              LABO                 310
IN TOTALE 6 NOMI NELL'ELENCO
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?5
IMPOSTARE IL CODICE SCELTO ?BE

ELENCO DEI NOMI CON CODICE BE

ROSSI                PIERO                BE                   100
ROSSINI              GIANNI               BE                   110

IN TOTALE 2 NOMI NELL'ELENCO
INDICARE IL TIPO DI RICERCA
1 = RICERCA DEL COGNOME
2 = RICERCA DEL COGNOME E NOME
3 = RICERCA DEL NOME
4 = RICERCA DEL NUMERO TELEFONICO
5 = ESTRAZIONE DEL NUMERO DI CODICE
?0

BREAK IN 230
```

Figura 4.22 (fine)

A partire dal momento in cui la ricerca è alfabetica, la ricerca del numero di telefono si può ottenere trasformando questi numeri in stringhe di caratteri da cui la linea 590

```
590 B$(I) = STR$ (T(I))
```

La sequenza della ricerca assomiglia molto a quella dell'esercizio 4.1 ma le permutazioni devono essere effettuate su C$, N$, S$, T e anche su B$.

Critiche al programma: Il grosso difetto di questo programma è la dimensione limitata dell'elenco dovuta al fatto che i dati sono inclusi nel programma e risiedono durante l'esecuzione nella memoria del calcolatore. Tuttavia con la stessa struttura generale è possibile costruire un nuovo programma che lavori con accesso diretto al file su disco, in questo caso è necessario porre attenzione a non aumentare inutilmente il numero di accessi al disco.

4.6 CONCLUSIONI

Gli esercizi orientati alla gestione non presentano particolari difficoltà dato che non richiedono l'accesso a file. E' tuttavia necessario lavorare con metodo.

Gli esercizi che fanno appello ai file non sono più difficili a livello di flow-chart ma la loro programmazione dipende dall'interprete BASIC impiegato, che è raramente compatibile per quanto concerne le istruzioni di trattamento dei file.

CAPITOLO 5

CALCOLI MATEMATICI

5.0 INTRODUZIONE

Il BASIC si presta bene all'esecuzione di semplici calcoli matematici. Generalmente i flow-chart sono molto semplici da realizzare e la programmazione presenta poche difficoltà.

Per contro, in certi casi, il propagarsi degli errori di arrotondamento provoca il raggiungimento di risultati poco precisi. Per premunirsi contro l'influenza di questi errori è possibile l'attuazione di diverse tecniche:

— prima scegliere degli algoritmi poco sensibili a questo tipo di errore, ciò non è sempre facile,
— in seguito programmare in modo che la perdita di precisione risulti più piccola possibile.

Nel contesto di un libro di esercizi, non è possibile dilungarsi su problemi di questa importanza, il lettore interessato potrà consultare delle pubblicazioni di analisi numerica. Nel frattempo, per attirare l'attenzione del lettore sulle possibili conseguenze, lo invitiamo a studiare attentamente i risultati ottenuti dal calcolo di "pi greco" con il metodo dei poligoni regolari inscritti.

5.1 DIVISIONE DI UN POLINOMIO PER $(x - s)$

Si consideri un polinomio p(x) di grado n di cui si conoscono i coefficienti:

$$P(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_{n-1} + a_n$$

Bisogna trovare il polinomio Q(x) di grado n-1 così che:

$$P(x) = (x - s)\, Q(x) + R$$

il resto R è una costante. Quindi:

$$Q(x) = b_0 x^{n-1} + \beta_1 \, {}^{xn-1} + \dots + b_{n-2} + b_{n-1}$$

si otterà:

$b_0 = a_0$
$b_1 = a_1 + sb_0$
$b_2 = a_2 + sb_1 \quad e\ R = an + \sigma\beta^{n+}{}_1$
$\beta_{\sim} = a_i + sb_i$
$b_{n-1} = a_{n-1} + sb_{n-2}$

Problema: Scrivere un programma che determini i coefficienti di Q(x) partendo dai coefficienti di P(x) e di (x—A1), essendo A1 ugualmente dato.

Soluzione: La parte relativa all'algoritmo è particolarmente semplice poichè per I che varia tra 1 e N—1 si può scrivere:

$b_i = a_i + sb_{i-1}$ ←——— Coefficiente di Q(x)

Coefficiente di P(x)

Si può comunque calcolare b_n con questa formula e fare:

$R = b_n$

Tutte le difficoltà consistono dunque nello scrivere bene le istruzioni di I/O.

Il programma di Fig.5.1 costituisce un esempio di ciò che si può fare. E' possibile evidentemente avere altre presentazioni dei risultati.

```
20  REM  DIVISIONE DI UN POLINOMIO PER X-A1
30  REM  N = GRADO DEL POLINIMIO
40  REM  IL VETTORE 'A' CONTIENE I COEFFICIENTI DI P(X)
50  REM  IL VETTORE 'B' TIENE IL CALCOLO DEI COEFFICIENTI DI Q(X)
60  REM  'R' E' IL RESTO DELLA DIVISIONE TRA P(X) E X-A1
100 DIM A(20),B(21)
105 REM  LEGGE UN INPUT
110 READ N,A1
115 PRINT "DIVISIONE DI P(X) PER (X-";A1;")
116 PRINT
120 FOR I = 1 TO N
130  READ A(I)
140  NEXT I
145 REM  CALCOLO DEI COEFFICIENTI DI Q(X)
150 B(0) = A(0)
160 FOR I = 1 TO N - 1
170  B(I) = A(I) + A1 * B(I - 1)
180  NEXT I
190 R = A(N) + A1 * B(N - 1)
195 REM  STAMPA DEI RISULTATI
200 PRINT "COEFFICIENTI DI P(X) ";
210 FOR I = 1 TO N
220  PRINT A(I);
230  NEXT I
240 PRINT
250 PRINT
```

```
260 PRINT "COEFFICIENTI DI Q(X) ";
270 FOR I = 0 TO N - 1
280  PRINT B(I);
290  NEXT I
300 PRINT
310 PRINT
320 PRINT "RESTO: ";R
330 STOP
340 DATA  6,1
350 DATA  3,2,-1,5,6,4,1
360 END
```

Figura 5.1

Esempio di esecuzione:

```
DIVISIONE DI P(X) PER (X-1)
COEFFICIENTI DI P(X) 32-1564
COEFFICIENTI DI Q(X) 0354915
RESTO: 19
BREAK IN 330
```

Figura 5.2

Si verifica facilmente che:

$$3x^6 + 2x^5 - x^4 + 5x^3 + 6x^2 + 4x + 1 =$$
$$(x\text{-}1)(3x^5 + 5x^4 + 4x^3 + 9x^2 + 15x + 19) + 2$$

Commenti:

1. In parecchi sistemi l'istruzione BASE 0 è destinata a mostrare che le variabili indicate hanno l'indice che parte da 0.

Secondo le norme standard ANSI l'istruzione BASE 0 dovrebbe essere scritta OPTION BASE 0.

2. Con un BASIC dove che gli indici devono obbligatoriamente partire da 1, è sufficiente sottrarre 1 dall'indice nel programma di figura 5.1.

3. La sequenza di stampa utilizzata può evidentemente essere modificata in funzione di una concreta applicazione. Si può notare tuttavia la necessità di inserire delle istruzioni PRINT per rendere più chiara la presentazione.

5.2 CALCOLO DI UN INTEGRALE DEFINITO

Tra i numerosi metodi che permettono di calcolare l'integrale definito di una funzione limitata su un intervallo ugualmente limitato, si possono citare:

— il metodo di SIMPSON
— il metodo di WEDDLE

che sono particolarmete semplici e saranno oggetto di due esercizi.

Metodo di Simpson: Con *n* pari, per calcolare:

$$S = \int_a^b f(x)\,dx$$

Si opera nel modo seguente:

— dividere l'intervallo a, b, in n intervalli con:

$$h = \frac{b - a}{n}$$

— calcolare poi:

$$S = \frac{h}{3}[f(x_0) + 4 f(x_1) + 2 f(x_2) + 4 f(x_3) + 2 f(x_4) + \cdots + 4 f(x_{n-1}) + f(x_n)]$$

con $x_0 = a$ e $x_n = b$.

Metodo di Weddle: Con n multiplo di 6, si calcola h come in precedenza, poi si valuta S con:

$$S = \frac{3h}{10}[f(a) + 5 f(a + h) + f(a + 2h) + 6 f(a + 3h) + f(a + 4h) + + 5 f(a + 5h) + f(b)]$$

se $n = 6$ o se $n = 12,18$, etc.

$$= \frac{3h}{10}[f(a) + 5 f(a + h) + f(a + 2h) + 6 f(a + 3h) + f(a + 4h) + + 5 f(a + 5h) + 2 f(a + 6h) + \cdots + f(b)]$$

Esercizio 1: Scrivere una subroutine che calcoli un integrale definito con il metodo di Simpson; scrivere un programma principale che utilizzi tale metodo per calcolare:

$$S = \int_{-1,95}^{+1,95} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Fare un calcolo con $n = 6, 12, 18, 24, 30$ senza tener conto del fatto che questa funzione è "pari".

Esercizio 2: Mantenendo lo stesso enunciato precedente si sostituisce al metodo di Simpson quello di Weddle.

Effettuare il calcolo con $n = 6, 12, 18, 24$.
Confrontare i risultati ottenuti.

Soluzione esercizio 1:

Per costruire un solo loop di calcolo definiamo due variabili S1 e S2 costituite rispettivamente dai termini di S con coefficiente 4 e 2.

$$S1 = f(x_1) + f(x_3) + \qquad + f(x_{n-1})$$
$$S2 = f(x_2) + f(x_4) + \qquad + f(x_{n-2})$$

sostituendo S si avrà

$$S = \frac{H}{3}(4\ S1 + 2\ S2 + f(a) + f(b))$$

Il flow-chart relativo alla parte di calcolo si costruisce ora facilmente (Fig.5.3) a condizione di ricordare che S1 deve normalmente contenere un termine superiore a quello di S2 e che si deve effettuare un loop comune, sarà necessario alla fine aggiungere $f(x_{n-1})$ che sarà stato dimenticato.

Soluzione esercizio 2:

Si farà riferimento al metodo dell'esercizio 1 con la differenza che in questo esercizio si faranno 6 somme parziali nel loop e, come precedentemente, una somma definita all'uscita del loop. Questo metodo allunga notevolmente il programma ma lo rende più semplice. Per questa ragione qui non viene raffigurato il flow-chart relativo.

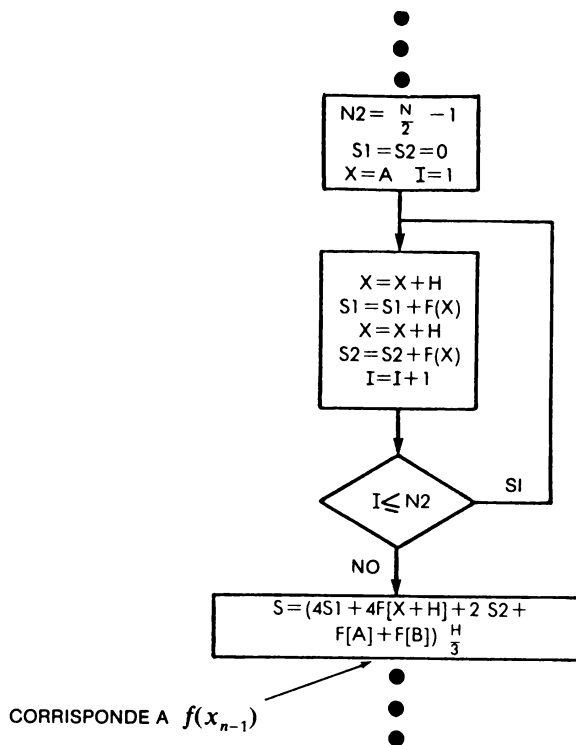



Figura 5.3

```
10 REM CALCOLO DI UN INTEGRALE CON IL METODO DI SIMPSON
50 PRINT "CALCOLO DI UN INTEGRALE DEFINITO CON IL METODO DI SIMPSON"
60 PRINT
70 PRINT " N  INTEGRALE"
80 PRINT
100 DEF FNE(X)=C*EXP(-X*X*0.5)
110 C=1/SQR(2*3.14159267)
112 A=-1.95
113 B=1.95
115 FOR N= 6 TO 30 STEP 6
120 GOSUB 3500
130 PRINT USING 200,N,S
140 NEXT N
150 STOP
200 :###      #.##########
3400 REM SOTTOPROGRAMMA DI CALCOLO DELL'INTEGRALE DEFINITO
3410 REM METODO DI SIMPSON
3420 REM 'H' RAPPRESENTA IL PASSO DI INTEGRAZIONE
3500 IF INT(N/2)<>N/2 THEN 3640
3510 N2=N/2-1
3520 H=(B-A)/N
3530 S1=0
3540 S2=0
3550 X=A
3560 FOR I=1 TO N2
3570 X=X+H
3580 S1=S1+FNE(X)
3590 X=X+H
3600 S2=S2+FNE(X)
3610 NEXT I
3620 S=(4*(S1+FNE(X+H))+2*S2+FNE(A)+FNE(B))*H/3
3630 RETURN
3640 PRINT " IMPOSSIBILE PERCHE' N E' DISPARI "
3650 STOP
3660 END
```

Esempio d'esecuzione

```
CALCOLO DI UN INTEGRALE DEFINITO CON IL METODO DI SIMPSON

N       INTEGRALE

6       .9485226178

12      .9488106717

18      .9488214551

24      .9488231313

30      .9488235761
```

Figura 5.4

```
10 REM CALCOLO DI UN INTEGRALE CON IL METODO DI WEDDLE
80 PRINT "CALCOLO DI UN INTEGRALE DEFINITO CON IL METODO DI WEDDLE"
90 PRINT
100 DEF FNE(X)=C*EXP(-X*X*0.5)
110 C=1/SQR(2*3.141592653)
120 A=-1.95
130 B=1.95
140 PRINT " N   INTEGRALE "
145 PRINT
150 FOR N=6 TO 30 STEP 6
160 GOSUB 4000
170 PRINT USING 200,N,S
175 NEXT N
180 STOP
200 :###    #.##########
4000 IF N-6*INT(N/6) <> 0 THEN 4200
4010 P=N/6
4020 X=A
4030 H=(B-A)/N
4040 S1=0
4050 S2=0
4060 S3=0
4070 S6=0
4100 FOR I=1 TO P
4110 S1=S1+FNE(X+H)
4120 S2=S2+FNE(X+2*H)
4130 S3=S3+FNE(X+3*H)
4140 S2=S2+FNE(X+4*H)
4150 S1=S1+FNE(X+5*H)
4160 S6=S6+FNE(X+6*H)
4165 X=X+6*H
4170 NEXT I
4180 S=.3*H*(FNE(A)-FNE(B)+5*S1+S2+6*S3+2*S6)
4190 RETURN
4200 PRINT "IMPOSSIBILE PERCHE' ";N;" NON E' MULTIPLO DI 6"
4210  STOP
4220 END
```

```
CALCOLO DI UN INTEGRALE DEFINITO CON IL METODO DI WEDDLE

N      INTEGRALE

6      .9501323783
12     .9488272534
18     .9488241485
24     .9488239269
30     .9488238929
```

Figura 5.5

Confronto dei risultati: Il risultato teorico esatto è 0.95. La tabella seguente mostra i risultati ottenuti con ciascuno dei metodi utilizzati.

N	SIMPSON	WEDDLE
6	.9485226178	.9501323783
12	.9488106717	.9488272534
18	.9488214551	.9488241485
24	.9488231313	.9488239269
30	.9488235761	.9488238929

Questa tabella mostra che il metodo di SIMPSON, meno preciso di quello di WEDDLE, fornisce risultati migliori quando si aumenta almeno per un certo tempo il numero dei passi per la durata dei calcoli.

Invece, i risultati migliori vengono ottenuti con l'altro metodo per N= 6 e N= 12 ed in seguito il risultato si allontana lentamente dal valore esatto. Questo può essere dovuto a cause diverse, per esempio per l'influenza degli errori di arrotondamento.

5.3 CALCOLO DI π CON IL METODO DEI POLIGONI REGOLARI

Si possono calcolare dei valori approssimativi di π assimilando il perimetro di un poligono regolare alla circonferenza di un cerchio inscritto o circoscritto, questo permetterà così di avvicinare con dei valori arrotondati per eccesso o per difetto.

Il poligono di partenza è quadrato. Partendo dal quadrato si possono calcolare due valori di π. Ad ogni iterazione, si darà il numero dei lati del poligono, e si effettuerà un nuovo calcolo:

- calcolo del lato,
- calcolo di π.

Quando l'intervallo che contiene π non diminuisce più per gli errori di arrotondamento, si arresteranno le iterazioni.

Analisi del problema: Si studierà ora il caso del poligono inscritto e di quello circoscritto:

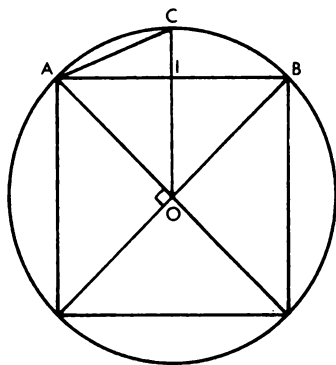
Poligono inscritto: Si considera un cerchio di raggio 1.

il lato del quadrato è dato da:
$AB^2 = OA^2 + OB^2$
da cui $AB = C = \sqrt{2}$

Figura 5.6

Si calcola ora la lunghezza di AC del nuovo lato:

$$AC^2 = IA^2 + IC^2 \quad \text{con} \quad IA = \frac{C}{2}$$

$$IC = OC - OI$$

OI è dato da $OI^2 = OA^2 - IA^2$

$$= 1 - \frac{C^2}{4}$$

Portandolo in AC^2 si ottiene:

$$AC^2 = \frac{C^2}{4} + 1 + 1 - \frac{C^2}{4} - 2\sqrt{1 - \frac{C^2}{4}}$$

$$= 2 - 2\sqrt{1 - \frac{C^2}{4}}$$

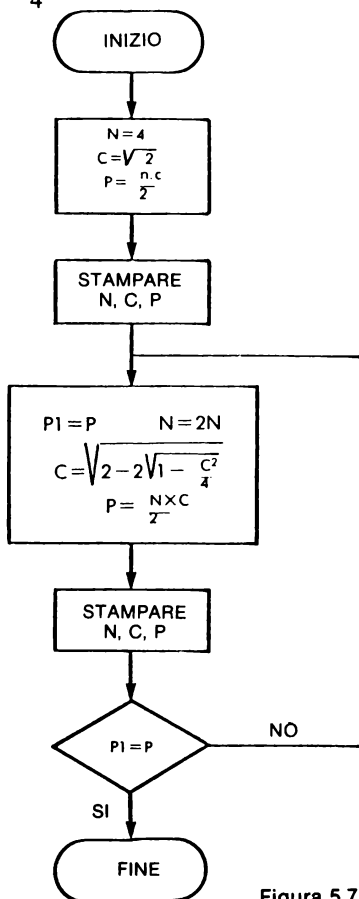



Figura 5.7

Partendo dalla lunghezza di AC si otterrà π con:

$$8 \cdot AC = 2\pi$$

in modo generale $\pi = \frac{n}{2} \times$ lato del poligono

Flow-chart provvisorio: Comporta una parte inizializzazione che corrisponde al quadrato e in seguito al calcolo iterattivo (Fig.5.5).

Poligono circoscritto: come prima si parte da un cerchio di raggio 1. Inizialmente il lato del quadrato è dato da:

$$AB = 2AJ \quad \text{con} \quad AJ = 1$$

da cui

$$AB = C = 2$$

La lunghezza del nuovo lato HK dev'essere determinata.

Per questo si calcoli:

$$IK^2 = AK^2 - IA^2$$

con $AK = \frac{C}{2} - IK$ poichè $IK = KJ$

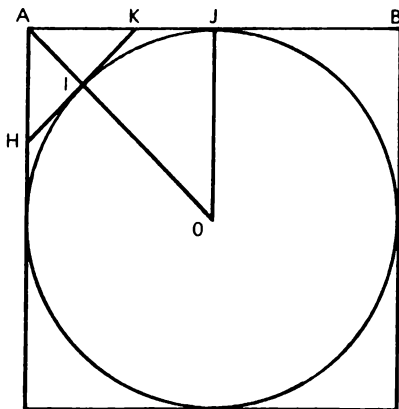



Figura 5.8

dunque

$$AK^2 = \frac{C^2}{4} - C \times IK$$

$$OA^2 = \frac{C^2}{4} + 1 \text{ dunque } \quad IA = \sqrt{1 + \frac{C^2}{4}} - 1$$

$$IA^2 = 2 + \frac{C^2}{4} - 2\sqrt{1 + \frac{C^2}{4}}$$

riportando in IK^2 otterremo:

$$IK^2 = \frac{C^2}{4} - C \,.\, IK + IK^2 - 2 - \frac{C^2}{4} + 2\sqrt{1 + \frac{C^2}{4}}$$

$$C \,.\, IK = 2\sqrt{1 + \frac{C^2}{4}} - 2$$

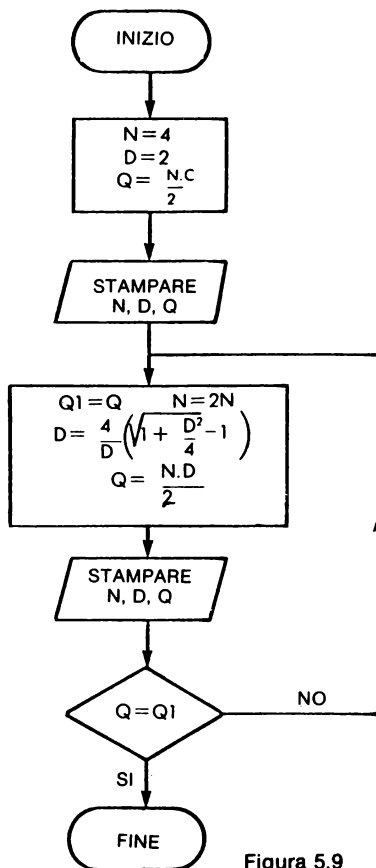



Figura 5.9

da cui la lunghezza del nuovo lato

$$HK = 2\,IK = \frac{4}{C}\left(\sqrt{1 + \frac{C^2}{4}} - 1\right)$$

Si avvicinerà π con $n\text{HK} = 2p$.

Flow-chart provvisorio: Con questo metodo di approccio dall'alto, si otterrà un nuovo flow-chart che assomiglierà molto al precedente.

Flow-chart definitivo: Si possono combinare i due flow-chart al fine di ottenere un flow-chart che permetta ad ogni iterazione di ridurre l'intervallo di scarto.

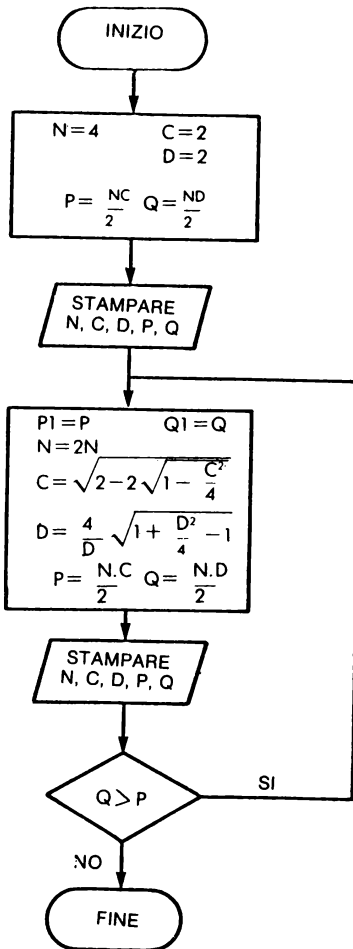



Figura 5.10

Partendo da questo flow-chart la programmazione non presenta particolari difficoltà. L'esperienza mostra che l'influenza degli errori di arrotondamento è notevole e che i risultati divergono assai velocemente. Si calcola quindi la differenza tra i limiti inferiori e superiori ottenuti da:

$$E = Q - P$$

e si arresta il programma quando E diviene negativo.

Inoltre si può calcolare π mezzi corrispondente alla media aritmetica di P e Q. Si constaterà che questa media si avvicina molto velocemente a π. Nonostante i calcoli vengano eseguiti con grande precisione, si constata che il valore esatto che comincia per

π: 3,14 15 9 2 6 5 3 5 8 9 7 9 3

non può essere avvicinato con questo metodo con precisione, poichè la nona cifra del valore medio d'intervallo è errata. Questo è dovuto al processo di calcolo che determina errori di arrotondamento che procedendo nel calcolo condurranno a risultati inesatti. Con un calcolatore avente una precisione inferiore, la precisione di π sarà ancora peggiore.

```
100 REM CALCOLO DI PI CON IL METODO DEL POLIGONO
101 REM INSCRITTO E DEL POLIGONO CIRCOSCRITTO
102 REM
110 N=4
120 C=SQR(2)
130 D=2
140 P=0.5*N*C
150 Q=0.5*N*D
152 M=0.5*(P+Q)
155 E=Q-P
160 PRINT USING 275
165 PRINT
170 PRINT USING 280,N,C,D,P,Q,M
180 P1=P
190 Q1=Q
200 N=2*N
210 C=SQR(2-2*SQR(1-0.25*C*C))
220 D=4*(SQR(1+0.25*D*D)-1)/D
230 P=0.5*N*C
240 Q=0.5*N*D
242 M=0.5*(P+Q)
245 E=Q-P
250 PRINT USING 280,N,C,D,P,Q,M
255 IF E<0 THEN 290
260 IF P1=P THEN 290
270 IF Q1<>Q THEN 180
275 :NUMERO       C           D          PI-INF          PI-SUF          PI-MEZZI
280 :###### #.####### #.####### #.########### #.########### #.###########
290 STOP
300 END
```

Figura 5.11

Esempio d'esecuzione:

NUMERO	C	D	PI-INF	PI-SUP	PI-MEZZI
4	1.4142136	2.0000000	2.82842712475	4.00000000000	3.41421356237
8	.7653669	.8284271	3.06146745892	3.31370849898	3.18758797895
16	.3901806	.3978247	3.12144515226	3.18259787807	3.15202151517
32	.1960343	.1969828	3.13654849055	3.15172490743	3.14413669899
64	.0981353	.0982537	3.14033115695	3.14411838524	3.14222477110
128	.0490825	.0490972	3.14127725093	3.14222362994	3.14175044043
256	.0245431	.0245449	3.14151380115	3.14175036914	3.14163208514
512	.0122718	.0122720	3.14157294028	3.14163208047	3.14160251037
1024	.0061359	.0061359	3.14158772485	3.14160250955	3.14159511720
2048	.0030680	.0030680	3.14159142158	3.14159511337	3.14159326747
4096	.0015340	.0015340	3.14159234176	3.14159326983	3.14159280579
8192	.0007670	.0007670	3.14159256943	3.14159277972	3.14159267458
16384	.0003835	.0003835	3.14159260738	3.14159273859	3.14159267298
32768	.0001917	.0001917	3.14159230381	3.14159186903	3.14159208642

Figura 5.12

5.4 RISOLUZIONE DI UN'EQUAZIONE CON DICOTOMIA

Data una equazione $f(x) = 0$ che ammetta almeno una radice nell'intervallo (a,b). Si supponga che la funzione sia continua e limitata in questo intervallo. Si supponga inoltre che $f(a)$ e $f(b)$ siano di segno contrario.

Soluzione: Il sottoprogramma deve poter funzionare correttamente qualunque siano le equazioni e i dati forniti. In particolare fornendo due punti A e B in modo che f(A) e f(B) siano dello stesso segno, il sottoprogramma dovrà accorgersene e fornire la segnalazione. Questo porta dunque a un primo controllo. Nell'esempio dato, si utilizza una variabile L alla quale si assegnano tre possibili valori:

— 1 se f(A) e f(B) sono dello stesso segno,
— 0 se, dopo una o più iterazioni, $f(x) <= $ E1,
— 1 se, dopo una o più iterazioni, vi è un intervallo di lunghezza inferiore o uguale a E.

Nell'esempio di soluzione si è supposto che l'utilizzatore abbia provveduto alla corretta impostazione dei valori di E1 e E2 che dovevano essere obbligatoriamente di segno positivo.

Per non modificare nel sottoprogramma i valori di A e B, si utilizzano due variabili ausiliarie A1 e B1 che rappresentano i limiti dell'intervallo. Si giunge così al flow-chart di Fig.5.13.

Nel programma di Fig.5.14 la divisione per 2 è stata sostituita da una moltiplicazione per 0.5 che è più rapida (linea 1050).

L'esempio trattato permette di ottenere il valore X= 7.9999945 quando la radice esatta è 8.

Algoritmo: A ciascuna iterazione si divide per due la lunghezza dell'intervallo.

1. Calcolare $x = \dfrac{a + b}{2}$

   $y = f(x)$

2. Se $f(x)$ e y sono dello stesso segno:

   porre $a = x$
   passare in 3

altrimenti

   porre $b = x$
   passare in 3

3. Provare se una delle seguenti condizioni è soddisfatta:

   $|y| \leqslant \varepsilon$
   $|b - a| \leqslant \eta$

con ε e η dati

— se nessuna condizione è soddisfatta tornare in 1 e proseguire le iterazioni,
— se una condizione è soddisfatta si arrestano le iterazioni.

Problema: Scrivere una subroutine che può ridurre una equazione secondo il metodo esposto.

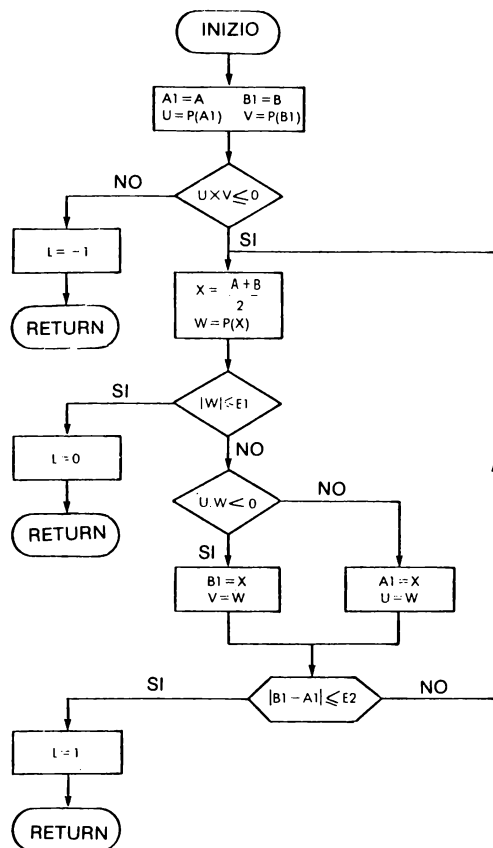



Figura 5.13

```
100  DEF  FN F(Z) = (Z + 1) * (Z - 8)
110 E1 = 0.0001:E2 = 0.001
120 A = 1.6:B = 100
130  GOSUB 1000
140  IF L =  - 1 THEN 300
150  PRINT "L= ";L
160  PRINT "SOLUZIONI TROVATE X= ";X;"    Y = ";W
170  PRINT
180  PRINT "INTERVALLO A = ";A1;"    B = ";B1
190  STOP
300  PRINT "NESSUNA SOLUZIONE; F(A) E F(B) STESSO SEGNO"
310  STOP
1000 A1 = A:B1 = B
1020 U =  FN F(A1)
1030 V =  FN F(B1)
1040  IF U * V > 0 THEN 1190
1050 X = 0.5 * (A1 + B1)
1060 W =  FN F(X)
```

Figura 5.14 (continua)

```
1070  IF  ABS (W) <  = E1 THEN 1170
1080  IF U * W < 0 THEN 1120
1090 A1 = X:U = W
1100  GOTO 1140
1120 B1 = X:V = W
1140  IF  ABS (B1 - A1) > E2 THEN 1050
1150 L = 1: RETURN
1170 L = 0: RETURN
1190 L =  - 1: RETURN
1200  END
```

Figura 5.14 fine

Esempio d'esecuzione

```
 RUN
L= 0
SOLUZIONI TROVATE X= 7.9999939    Y = -5.49014286E-05
INTERVALLO A = 7.99924317    B = 8.00074464
BREAK IN 190
```

Figura 5.15

5.5 VALORE NUMERICO DI UN POLINOMIO

Si cerca di calcolare il valore numerico di un polinomio P(x) di cui si conoscono i coefficienti e i valori della variabile. Per questo si utilizza il metodo di Höerner che minimizza il numero delle operazioni. Per la valutazione del valore di P(x) dato da:

$$P(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$$

si calcola:

$$P : (\ldots (((a_0 x + a_1) x + a_2) x + a_3) x + \cdots + a_{n-1}) x + a_n$$

Nel programma di Fig.5.16, si è utilizzato il secondo metodo che è più semplice. Il sottoprogramma comporta 5 istruzioni (linee 1000-1040) compresa l'istruzione RETURN.

Problema: Costruire un programma che per mezzo di un sottoprogramma valuti P(x) con i valori di x dati dal programma principale.

Soluzione: La formula data significa che sarà necessario fare i seguenti calcoli:

$P = a_0 x + a_1$

Poi :

$P = (P + a_2) x$

Poi :

$P = (P + a_3) x$

e così di seguito fino a:

$P = (P + a_{n-1}) x$

fino a

$P = a_0$

Poi :

$P = Px + a_1$

Poi :

$P = Px + a_2$

e così di seguito fino a:

$P = Px + a_n$

infine:

$$P = P + a_n$$

Da qui si può vedere come può essere utilizzata la seguente iterazione:

```
100  REM  VALORE NUMERICO DI UN POLINOMIO SECONDO LO SCHEMA DI HOERNER
110  DIM A(100)
115  PRINT "IMPOSTARE IL GRADO DEL POLINOMIO ";
120  INPUT N
130  PRINT "IMPOSTARE I ";N + 1;" COEFFICIENTI IN ORDINE DECRESCENTE"
140  FOR I = 0 TO N
150  INPUT A(I)
160  NEXT I
170  PRINT "IMPOSTARE IL VALORE DI X PER IL VALORE DEL POLINOMIO";
190  INPUT X
200  IF X = 0 THEN  STOP
210  GOSUB 1000
220  PRINT
230  PRINT "VALORE DEL POLINOMIO = ";P
240  PRINT
250  GOTO 170
990  REM  VALUTAZIONE DEL POLINOMIO CON IL METODO DI HOERNER
1000 P = A(0)
1010  FOR I = 1 TO N
1020 P = P * X + A(I)
1030  NEXT I
1040  RETURN
1050  END
```

Figura 5.16

```
IMPOSTARE IL GRADO DEL POLINOMIO ?2
IMPOSTARE I 3 COEFFICIENTI IN ORDINE DECRESCENTE
?1
?1
?1
IMPOSTARE IL VALORE DI X PER IL VALORE DEL POLINOMIO?2
VALORE DEL POLINOMIO = 7
IMPOSTARE IL VALORE DI X PER IL VALORE DEL POLINOMIO?3
VALORE DEL POLINOMIO = 13
IMPOSTARE IL VALORE DI X PER IL VALORE DEL POLINOMIO?10
VALORE DEL POLINOMIO = 111
IMPOSTARE IL VALORE DI X PER IL VALORE DEL POLINOMIO?0
BREAK IN 200
```

Figura 5.17

5.6 CONCLUSIONI

Questi esercizi mostrano che la soluzione dei problemi matematici iterativi (calcolo di integrali definiti, risoluzione di equazioni, etc.) presenta pochi problemi dal punto di vista della programmazione. Del resto il flow-chart è più semplice da impostare rispetto agli esercizi di aritmetica presentati nel Capitolo 2. Inoltre, per questo tipo di problemi, è necessario poter effettuare un gran numero di operazioni di calcolo con precisione e il microcalcolatore si rivela adatto a questo genere di calcolo.

Qualche volta si pone per contro il problema della "validità dei risultati", poichè gli errori di metodo e gli errori di arrotondamento possono avere un'influenza seria soprattutto quando i calcoli diventano più complessi. Il lettore interessato a questi problemi è invitato a leggere dei testi "di analisi numerica", che trattano metodi di calcolo per microcalcolatori e dell'influenza degli errori.

CAPITOLO 6

CALCOLI FINANZIARI

6.1 PREVISIONE DI CIFRE DI AFFARI

In questi esercizi si suppone noto il tasso di crescita e si cerca di prevedere l'evoluzione della cifra di affari. I due programmi, facili da realizzare, sono raramente utilizzabili nella pratica, ma possono servire da base alla redazione di programmi specifici.

Primo problema: Un'impresa realizza una data cifra di affari C e prevede un tasso di crescita T per gli N anni futuri. Determinare la cifra di affari prevista. I dati sono:

A = anno d'inizio,
C = cifra d'affari per l'anno A,
T = tasso di crescita espresso in percentuale (%),
N = numero degli anni per i quali si desidera la previsione.

Esempio:

A = 1978
C = 20.000.000
T = 20 %
N = 5

Soluzione: Le sole difficoltà che s'incontrano sono quelle relative alla presentazione. Si dovrà tener conto del fatto che il tasso si esprime in percentuale (%) e dà luogo a un coefficiente moltiplicativo T1 dato da:

$$T1 = 1 + \frac{T}{100}$$

```
100  PRINT "PREVISIONE DELLE CIFRE D'AFFARI"
110  PRINT
120  PRINT "IMPOSTARE ANNO E CIFRA INIZIALE ";
130  INPUT A,C
140  PRINT
150  PRINT "IMPOSTARE TASSO DI CRESCITA ";
160  INPUT T
170  PRINT
180  PRINT "IMPOSTARE NUMERO ANNI DI PREVISIONE ";
190  INPUT N
200  PRINT
210  PRINT "ANNI","CIFRE D'AFFARI"
220  PRINT
230  PRINT A,C
240  T1 = 1 + 0.01 * T
250  FOR I = 1 TO N
260    A = A + 1
270    C = C * T1
280    PRINT A,C
290    NEXT I
300  END
```

Figura 6.1

Esempio d'esecuzione:

```
PREVISIONE DELLE CIFRE D'AFFARI

IMPOSTARE ANNO E CIFRA INIZIALE ?1978,1220

IMPOSTARE TASSO DI CRESCITA ?13

IMPOSTARE NUMERO ANNI DI PREVISIONE ?6

ANNI            CIFRE D'AFFARI

1978            1220
1979            1378.6
1980            1557.818
1981            1760.33434
1982            1989.1778
1983            2247.77092
1984            2539.98114
```

Figura 6.2

Secondo problema: Si parte questa volta da due dati di base: un volume di vendita e una cifra d'affari. Si prevede un aumento in volume di Q% e un rialzo annuale dei prezzi di T%.

Sarà necessario ora prevedere per gli N anni futuri l'evoluzione in volume di vendita e l'evoluzione della cifra d'affari.

I dati sono:

Anno d'inizio	A
Volume	V
Cifre d'affari	C
Aumento del volume	Q (in percentuale)
Rialzo dei prezzi	T (in percentuale)
Numero degli anni	N

Soluzione: Si potrà partire dal programma precedente. Si dovrà questa volta, tuttavia, tener conto dei due rialzi.

```
100  PRINT "IMPOSTARE ANNO, VOLUME E CIFRE D'AFFARI ";
110  INPUT A,V,C
120  PRINT "IMPOSTARE L'AUMENTO IN VOLUME E RIALZO DEL PREZZO ";
130  INPUT Q,T
140  PRINT "IMPOSTARE NUMERO DEGLI ANNI DI PREVISIONE ";
150  INPUT N
160  Q1 = 1 + .01 * Q
170  T1 = Q1 * (1 + .01 * T)
180  PRINT
190  PRINT "ANNO","VOLUME","CIFRE D'AFFARI"
200  PRINT
210  PRINT A,V,C
220  FOR I = 1 TO N
230    A = A + 1
240    V = V * Q1
250    C = C * T1
260    PRINT A,V,C
270    NEXT I
280  END
```

Figura 6.3

Esempio di esecuzione:

```
IMPOSTARE ANNO, VOLUME E CIFRE D'AFFARI ?1978,100,15000
IMPOSTARE L'AUMENTO IN VOLUME E RIALZO DEL PREZZO ?5,10
IMPOSTARE NUMERO DEGLI ANNI DI PREVISIONE ?6

ANNO           VOLUME         CIFRE D'AFFARI

1978           100            15000
1979           105            17325
1980           110.25         20010.375
1981           115.7625       23111.9831
1982           121.550625     26694.3405
1983           127.628156     30831.9633
1984           134.009564     35610.9176
```

Figura 6.4

Nota: In pratica sarà interessante utilizzare le istruzioni PRINT USING e IMAGE al fine di ottenere una migliore presentazione dei risultati.

6.2 RIMBORSO DI UN CAPITALE

Il rimborso di un capitale può essere fatto in molti modi. Qui vengono presentati due metodi relativamente semplici e vengono date delle indicazioni concernenti possibili variazioni.

6.2.1 Primo metodo di rimborso

Un capitale C è rimborsato in N anni. Alla fine di ogni anno viene rimborsata la

stessa frazione del capitale C/N e gli interessi corrispondenti alla frazione del capitale che non è stato ancora rimborsato.

Esempio: capitale C
numero degli anni N
tasso T

alla fine del primo anno si rimborsa: $C/N + C \times T$

alla fine del secondo anno si rimborsa: $C/N + C - C/N \times T$

e così di seguito...

Scrivere un programma che effettui questi calcoli e che stampi anche la somma dei rimborsi.

Soluzione: Si utilizzeranno prima le seguenti variabili:

R1 = C/N rappresenta la parte del capitale rimborsato ogni anno,
I importo dell'interesse versato ogni anno,
R2 importo del rimborso annuale,
(R2 = R1 + I)

Per calcolare I, è necessario conoscere il capitale sul quale vanno conteggiati gli interessi. Sia Y questo capitale; alla partenza Y è uguale a C e ogni anno, è necessario togliere da Y la quantità R1 rimborsata. Questo conduce allo schema seguente:

Calcoli preliminari
- T = T / 100 poichè T è dato in %
- R1= C/N
- Y = C

poi per ciascun anno
- $I = Y \times T$
- R2 = R1 + I
- Y = Y — R1

Per calcolare la somma totale rimborsata, vi sono due possibili metodi:

— effettuare il cumulo dei rimborsi: all'inizio Q = 0,
effettuare il cumulo a ogni rimborso Q = Q + R2
— effettuare il cumulo degli interessi: all'inizio Q = C,
a ogni rimborso effettuare il cumulo Q = Q + I,

dal punto di vista teorico i due metodi sono identici. Ma in pratica, il primo metodo permette di tener conto con maggior precisione degli arrotondamenti.

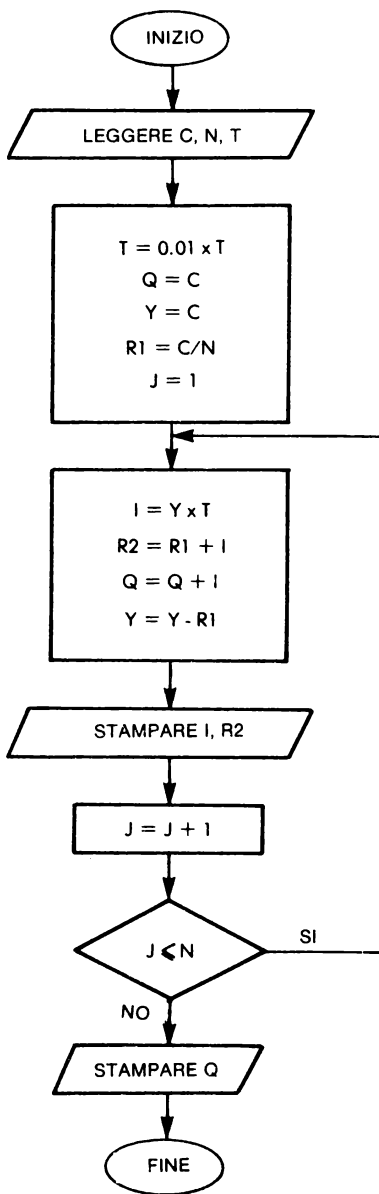



Figura 6.5

```
100  REM  PROGRAMMA DI CALCOLO ANNUALITA' . IN
110  REM  QUESTO PROGRAMMA, IN CIASCUN ANNO
115  REM  SI RIMBORSA LA STESSA FRAZIONE DEL CAPITALE.
120  INPUT "IMPOSTARE CAPITALE, TASSO E NUMERO ANNI ";C,T,N
130  T = T * 0.01
140  Q = C:Y = C
150  R1 = C / N
160  PRINT
170  PRINT "NUMERO  INTERESSE  MONTANTE"
180  FOR J = 1 TO N
190   I = Y * T
200   R2 = R1 + I
210   Q = Q + I
220   Y = Y - R1
230   PRINT J; TAB( 9);I; TAB( 20);R2
240   NEXT J
250  PRINT : PRINT "SOMMA TOTALE RIMBORSATA = ";Q
260  END
```

Figura 6.6

Esempio di esecuzione:

```
IMPOSTARE CAPITALE, TASSO E NUMERO ANNI 10000,10,10

NUMERO           INTERESSE        MONTANTE
1                1000             2000
2                900              1900
3                800              1800
4                700              1700
5                600              1600
6                500              1500
7                400              1400
8                300              1300
9                200              1200
10               100              1100

SOMMA TOTALE RIMBORSATA = 15500


IMPOSTARE CAPITALE, TASSO E NUMERO ANNI 10000,12,7

NUMERO  INTERESSE  MONTANTE
1         1200                   2628.57143
2         1028.57143                   2457.14286
3         857.142857                   2285.71429
4         685.714286                   2114.28571
5         514.285714                   1942.85714
6         342.857143                   1771.42857
7         171.428571                   1600

SOMMA TOTALE RIMBORSATA = 14800
```

Figura 6.7

6.2.2 Rimborso con mensilità costanti

Un capitale C è stato prestato a un tasso annuale T1. Ogni mese si rimborsa la stessa somma. Sapendo che con N mensilità si deve rimborsare tutto, calcolare l'importo della mensilità e su domanda dell'utilizzatore la ripartizione del capitale rimborsato, e di quanto pagato per gli interessi.

Per questo si opererà nel seguente modo:

— calcolo del tasso equivalente T1 (corrisponde al tasso mensile). Questo tasso T1 è definito con $(1 + T1)^{12} = 1 + T$.

da cui, estrapolando, si ricava:

$T1 = (1 + T) - 1$

se T è dato in percentuale, sarà necessario a priori dividerlo per cento.

E' da notare che sovente le banche applicano una formula che è loro più favorevole:

$$T1 \text{ mensile} = \frac{T \text{ annuale}}{12}$$

— calcolo dell'importo della mensilità per $M = C \dfrac{T1(1 + T1)^N}{(1 + T1)^N - 1}$

— calcolo eventuale del dettaglio.

La parte del capitale rimborsata si ottiene con un ragionamento molto semplice: per la prima mensilità, la quota interesse è: $C \times T1$,
dunque il capitale rimborsato è: $M - C \times T1$,
il capitale restante $C - (M - C \times T1)$ servirà per il calcolo degli interessi della seconda mensilità e così di seguito.

Enunciazione del problema: Scrivere un programma che:

— legga i dati:
  — capitale,
  — tasso annuale espresso in percentuale
  — numero delle mensilità;

— faccia i calcoli e stampi:
  — tasso equivalente,
  — importo delle mensilità,
  — somma totale rimborsata;

— domandi all'utilizzatore se desidera il dettaglio dei rimborsi e se si, i numeri della prima e dell'ultima mensilità per le quali si desidera il dettaglio.

Soluzione: L'inizio del programma deriva dall'enunciazione e dalle formule date a condizione di:

— leggere T in percentuale,

— calcolare T = T/100,
— verificare il valore T1 ottenuto, ma stampare 100 × T1 per darlo in percentuale.

Per la seconda parte verrà disegnato un flow-chart nel quale A e B rappresentano i numeri rispettivamente della prima e dell'ultima mensilità per le quali si desiderano i dettagli.

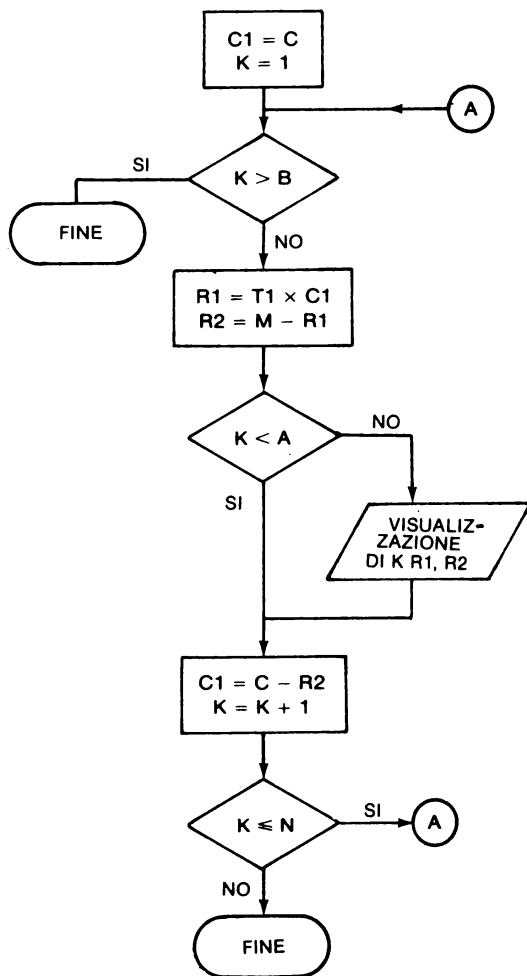



Figura 6.8

C1 rappresenta la parte del capitale non ancora rimborsato. In questo flow-chart (Fig.6.8), è stato previsto un loop per K da 1 a N per eventuali estensioni. Se ci si

limita al problema esposto è sufficiente prevedere un loop per K che varia da 1 a B, il che permette di sopprimere il test K > B.

```
100  REM  CALCOLO DELLE MENSILITA' PER RIMBORSARE UN CAPITALE
110  INPUT "MONTANTE DEL CAPITALE PRESTATO: ";C
120  INPUT "TASSO ANNUALE IN %               : ";T
130  INPUT "NUMERO DELLE MENSILITA'          : ";N
140  T1 = (1 + T / 100) ^ (1 / 12) - 1
150  M = C * T1 / (1 - (1 + T1) ^ ( - N))
160  PRINT
170  PRINT "TASSO EQUIVALENTE = ";100 * T1;" % "
180  PRINT
190  PRINT "MONTANTE DI OGNI MENSILITA' = "M
200  PRINT
210  PRINT "SOMMA TOTALE RIMBORSATA     = "M * N
220  PRINT
230  INPUT "VOLETE IL DETTAGLIO DI ALCUNE MENSILITA' ? ";R$
240  IF R$ <  > "SI" THEN  STOP
250  INPUT "NUMERI DELLE MENSILITA' ? ";W,B
260  PRINT "NUMERO","INTERESSE","AMMORTAMENTO"
270  PRINT :C1 = C
280  FOR K = 1 TO N
290   IF K > B THEN 350
300   R1 = T1 * C1:R2 = M - R1
310   IF K < W THEN 330
320   PRINT K,R1,R2
330   C1 = C1 - R2
340   NEXT K
350  END
```

Figura 6.9

Esempio di esecuzione:

```
MONTANTE DEL CAPITALE PRESTATO: 33322
TASSO ANNUALE IN %             : 6
NUMERO DELLE MENSILITA'        : 144
TASSO EQUIVALENTE = .486755045 %
MONTANTE DI OGNI MENSILITA' =                          322.438624
SOMMA TOTALE RIMBORSATA     =                         46431.1619
VOLETE IL DETTAGLIO DI ALCUNE MENSILITA' ? SI
NUMERI DELLE MENSILITA' ? 50,55
NUMERO         INTERESSE                   AMMORTAMENTO
50        119.151941                  203.286683
51        118.162433                  204.276191
52        117.168108                  205.270516
53        116.168943                  206.269681
54        115.164915                  207.273709
55        114.156                     208.282624

MONTANTE DEL CAPITALE PRESTATO: 60000
TASSO ANNUALE IN %             : 8.5
NUMERO DELLE MENSILITA'        : 180
TASSO EQUIVALENTE = .682149362 %
MONTANTE DI OGNI MENSILITA' =                          579.845213
SOMMA TOTALE RIMBORSATA     =                        104372.138
VOLETE IL DETTAGLIO DI ALCUNE MENSILITA' ? NO
BREAK IN 240

MONTANTE DEL CAPITALE PRESTATO: 30000
TASSO ANNUALE IN %             : 8
NUMERO DELLE MENSILITA'        : 120
TASSO EQUIVALENTE = .643403037 %
MONTANTE DI OGNI MENSILITA' =                          359.57257
SOMMA TOTALE RIMBORSATA     =                         43148.7084
VOLETE IL DETTAGLIO DI ALCUNE MENSILITA' ? NO
BREAK IN 240
```

Figura 6.10

6.3 CALCOLO DEL TASSO DI CRESCITA

Questo esercizio non dev'essere affrontato se non prima dell'esercizio sul calcolo della regressione lineare (9.3). Si conoscono le cifre d'affari degli anni precedenti di un'impresa. Si pensa che la cifra d'affari segua una legge matematica del tipo:

$$C(1 + \tau)^i$$

τ rappresenta il tasso di crescita e i l'anno corrente.

Il problema è determinare C e τ di prevedere le cifre d'affari per gli anni futuri. Si potrà limitare la previsione a 5 anni.

Analisi matematica: Per risolvere questo problema si adotterà un artificio sul piano matematico al fine di renderlo più facilmente risolvibile. Si cercherà di determinare C e τ in modo da minimizzare non:

$$\sum_i (c(1 + \tau)^i - y_i)^2$$

ma

$$Q = \sum_i (\text{Log}[c(1 + \tau)^i] - \text{Log}\, y_i)^2$$

$$= \sum_i (\text{Log}\, c + i\, \text{Log}(1 + \tau) - \text{Log}\, y_i)^2$$

poniamo:

$$\text{Log}\, C = B \qquad \text{Log}\, y_i = z_i$$
$$\text{Log}\,(1 + \tau) = A$$

E' necessario minimizzare la quantità Q data con:

$$Q = \sum (B + iA - z_i)^2$$

Questo ci riporta al problema della regressione lineare esposta nel Capitolo 9.3.

Problema: Partendo dal programma dato nel Capitolo 9.3, costruire un programma che calcoli il tasso di crescita τ ed effettui una previsione per i cinque anni futuri.

In questo programma τ sarà rappresentato dalla variabile TO. Si supporrà che gli anni siano immagazzinati in una tabella T e le cifre d'affari in una tabella X.

Soluzione: Questo programma comporta tre parti distinte:

— lettura dei dati N, tabelle T(I) e X(I) e costruzione della tabella Y come: Y(I) = Log (X(I))

— richiamo della subroutine che effettua la regressione lineare e calcola dei coefficienti C e TO. Partendo da A e B dati dalla subroutine, si ottengono questi coefficienti:

$$C = e^B \text{ et } TO = e^A - 1$$

— stampa di TO e dei risultati:
  — Per ogni anno conosciuto, cifra d'affari vera e cifra d'affari stimata.
  — Per ognuno dei cinque anni futuri, unicamente la cifra d'affari stimata.

Per evitare la stampa dei decimali inutili si rimpiazzerà la stampa delle cifre d'affari stimate Z con:

INT (100 × Z) / 100

Il flow-chart generale riportato in Fig.6.11, è quindi molto semplice.

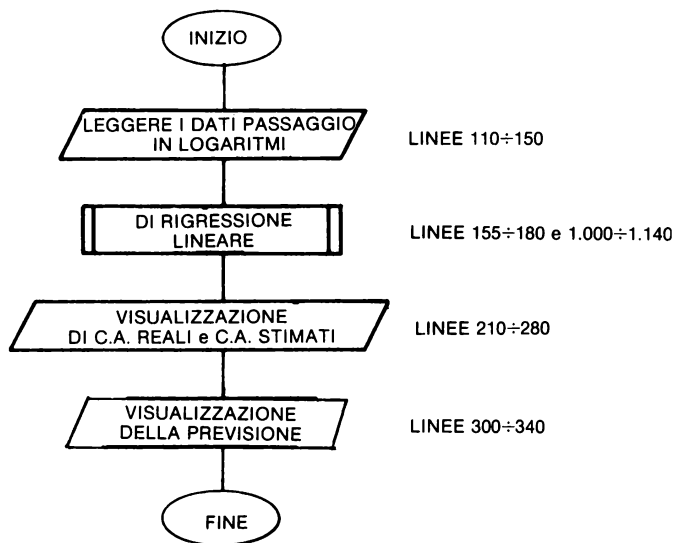



Figura 6.11

Il programma di Fig.6.12 costituisce un esempio di quanto si può scrivere, sarà possibile migliorarlo stampando:

— coefficiente di correlazione
— un intervallo stima per la previsione che sarà utile ma complicherà il programma.

Attenzione: Una tale previsione non può essere utilizzata in pratica senza precauzioni. In effetti la cifra d'affari di un'impresa dipende da numerosi fattori in modo particolare, dalla congiuntura economica e dalla concorrenza, e questi fattori possono influenzare notevolmente le realizzazioni future.

```
100  DIM T(15),X(15),Y(15)
110  READ N
120  FOR I = 1 TO N
130   READ T(I),X(I)
140   Y(I) =  LOG (X(I))
150   NEXT I
155 N0 = T(1)
160  GOSUB 1000
170 C =  EXP (B)
180 T0 =  EXP (A) - 1
190 PRINT "TASSO STIMATO = ";100 * T0
200 PRINT
210 T2 = 1 + T0
220 PRINT "ANNO","ATTUALE"."PREVISTO"
230 PRINT
240 Z = C
260  FOR I = 1 TO N
280  PRINT T(I),X(I), INT (100 * Z) / 100
285   Z = Z * T2
290  NEXT I
300 FOR I = 1 TO 5
310  T3 = T(N) + I
320  Z = Z * T2
330  PRINT T3, INT (100 * Z) / 100
340  NEXT I
400 DATA  6
410 DATA  1975,99.2,1976,110,1977,121.3
420 DATA  1978,133.1,1979,146.3
430 DATA  1980,160
500 STOP
1000 U1 = 0
1010 U2 = 0
1020 V1 = 0:V2 = 0
1040 W = 0
1050 FOR I = 1 TO N
1055  T4 = T(I) - N0
1060  U1 = U1 + T4
1070  V1 = V1 + Y(I)
1080  U2 = U2 + T4 * T4
1090  V2 = V2 + Y(I) * Y(I)
1100  W = W + T4 * Y(I)
1110  NEXT I
1120 A = (W - U1 * V1 / N) / (U2 - U1 * U1 / N)
1130 B = (V1 - A * U1) / N
1140 RETURN
1150 END
```

Figura 6.12

Esempio di esecuzione:

TASSO STIMATO = 10.0084655

ANNO	ATTUALE	PREVISTO
1975	99.2	99.76
1976	110	109.75
1977	121.3	120.73
1978	133.1	132.82
1979	146.3	146.11
1980	160	160.73
1981	194.52	
1982	213.99	
1983	235.4	
1984	258.96	
1985	284.88	

BREAK IN 500

Figura 6.13

6.4 CALCOLO DELL'AMMONTARE DELLE IMPOSTE

Supponiamo di voler calcolare a quanto ammontano le tasse dovute da un semplice salariato e supponiamo che la legge fiscale preveda quanto segue:

— Determinazione del reddito imponibile poi calcolo del quoziente familiare. Questo reddito imponibile è definito da R = 0,72 × reddito netto + pensione — deduzioni.

Il quoziente familiare è dato da: Q = R/N, con N il numero dei componenti la famiglia.

— Calcolo dell'importo d'imposta; sono possibili due metodi:
 — l'applicazione di una formula diretta,
 — il calcolo per passi successivi utilizzando la tecnica degli scaglioni tariffari,
  — calcolo dell'imposta relativa ad ogni scaglione
  — cumulo
  — poi moltiplicazione per il numero dei componenti

— Si deducono in seguito dall'importo lordo alcune eventuali voci per ottenere l'importo netto.

Per i redditi incassati nel corso dell'anno la tabella di Fig.6.14 indica gli ipotetici scaglioni tariffari.

REDDITO IN MILIONI	ALIQUOTE
7.925	0%
8.300	5%
9.925	10%
15.700	15%
20.625	20%
25.925	25%
31.350	30%
36.175	35%
62.600	40%
86.125	45%
105.950	50%
125.050	55%
oltre	60%

Figura 6.14

Esempio: Supponiamo che una famiglia composta da tre persone abbia incassato redditi per 130.000.000 di lire. Questa famiglia può dedurre 5.000.000 di lire rappresentati dagli interessi di un prestito contratto per l'acquisto di una casa.

Reddito imponibile: 0,72 × 130.000.000 — 5.000.000 = 88.600.000

Quoziente familiare: 88.600.000/3 = 29.533.000

Importo d'imposta:

(8.300.000 - 7.925.000) × 5 / 100 = 18.750
(9.925.000 - 8.300.000) × 10 / 100 = 162.500
(15.700.000 - 9.925.000) × 15 / 100 = 866.250
(20.625.000 - 15.700.000) × 20 / 100 = 985.000
(25.925.000 - 20.625.000) × 25 / 100 = 1.325.000
(29.533.000 - 25.925.000) × 30 / 100 = 1.082.400

TOTALE 4.439.900

Questo importo moltiplicato per i componenti dà: 4.439.900 × 3 = 13.319.700

Da questa somma si potrà dedurre un eventuale "avere fiscale". Si constati che questa famiglia raggiunge lo scaglione tariffario del 30%.

Problema: Il fine di questo esercizio è evidentemente la scrittura di un programma che, partendo da alcuni dati, calcoli l'importo dell'imposta sul reddito. Si propone di seguire i passi sotto indicati:

- precisare il metodo in modo da ridurre il numero degli scaglioni (scaglione con tasso 0) al fine di ottenere 12 scaglioni utili,
- costruire in seguito il flow-chart di dettaglio unicamente per la parte calcolo dell'imposta,
- scrivere un programma che chiama un sottoprogramma per la lettura degli scaglioni e delle percentuali e un altro sottoprogramma per il calcolo del montante d'imposta.

Soluzione: Siano F e T due tabelle contenenti rispettivamente i valori "più bassi" degli scaglioni tariffari e le relative aliquote. Non si è interessati allo scaglione 0% quindi si avrà:

F(1) = 7 925　　　　F(12) = 125 050
T(1) = 0,05　　　　T(12) = 0,6

Se Q ⩽ F(1) nessuna imposta,
se F(1) < Q ⩽ F(2) si calcolerà S = (Q — F(1)) × T(1)

— in generale, se $F(K-1) < Q \leqslant F(K)$ si calcolerà:

$$S = (Q - F(K-1)) \times T(K-1) + \sum_{I=2}^{K-1} (F(I) - F(I-1)) \times T(I-1)$$

Questo porta alla parte di flow-chart in Fig.6.15.

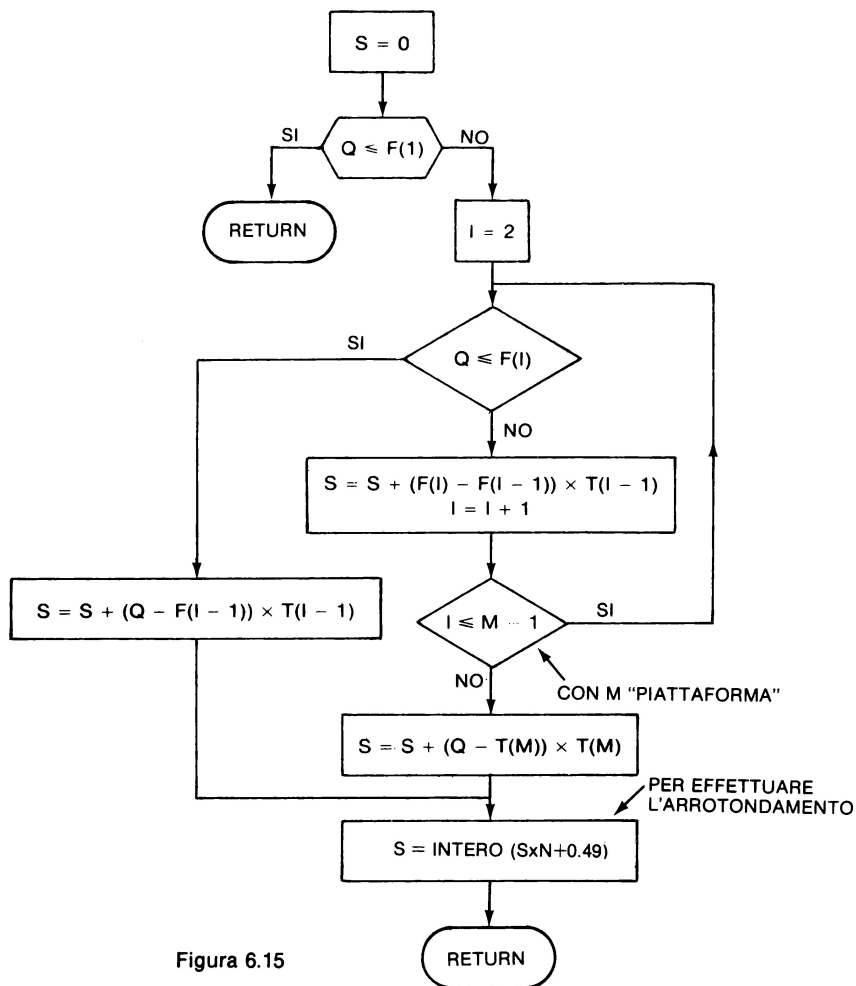



Figura 6.15

E' necessario ora scrivere un programma che sia facilmente adattabile alla modifica degli scaglioni:

— il sottoprogramma di lettura (linee 1000 - 1170) legge il numero degli scaglioni M poi le coppie F(I) e T(I)

— il sottoprogramma di calcolo (linee 1200 - 1290) corrisponde al flow-chart di Fig.6.15.

Il programma principale è quindi molto corto: pone le domande necessarie e verifica che i dati impostati siano corretti. Calcola in seguito il reddito imponibile e il quoziente familiare poi chiama il sottoprogramma per il calcolo delle tasse.

```
80 DIM F(15),T(15)
90 GOSUB 1000
100 INPUT "IMPOSTARE IL REDDITO NETTO ";R1
110 IF R1 < 0 THEN 100
120 INPUT "IMPOSTARE IL MONTANTE DELLA PENSIONE ";P
130 IF P < 0 THEN 120
140 INPUT "IMPOSTARE IL MONTANTE DELLE DEDUZIONI ";D
150 IF D < 0 THEN 140
160 R = .72 * R1 + P - D
170 INPUT "IMPOSTARE IL N. DEI COMPONENTI ";N
180 IF N < 0 OR N < > INT (N) THEN 170
190 Q = R / N
200 PRINT : PRINT "REDDITO IMPONIBILE = ";R
205 PRINT "QUOZIENTE FAMILIARE = ";Q
210 GOSUB 1200
220 PRINT "MONTANTE DELLE IMPOSTE = ";S
230 STOP
990 REM  SOTTOPROGRAMMA DI LETTURA DEGLI SCAGLIONI E DEI TASSI
1000 READ M
1010 FOR I = 1 TO M
1020  READ F(I),A:T(I) = .01 * A
1030  NEXT I
1040 RETURN
1050 DATA  12
1060 DATA  7925,5
1070 DATA  8300,10
1080 DATA  9925,15
1090 DATA  15700,20
1100 DATA  20625,25
1110 DATA  25925,30
1120 DATA  31350,35
1130 DATA  36175,40
1140 DATA  62600,45
1150 DATA  86125,50
1160 DATA  105950,55
1170 DATA  125050,60
1200 S = 0: IF Q <  = F(1) THEN  RETURN
1210 FOR I = 2 TO M - 1
1220  IF Q < F(I) THEN 1270
1230  S = S + (F(I) - F(I - 1)) * T(I - 1)
1240  NEXT I
1250 S = S + (Q - F(M)) * T(M)
1260 GOTO 1280
1270 S = S + (Q - F(I - 1)) * T(I - 1)
1280 S =  INT (N * S + .49)
1290 RETURN
```

Figura 6.16

Esempio di esecuzione:

```
IMPOSTARE IL REDDITO NETTO 1000000
IMPOSTARE IL MONTANTE DELLA PENSIONE 0
IMPOSTARE IL MONTANTE DELLE DEDUZIONI 0
IMPOSTARE IL N. DEI COMPONENTI 1
REDDITO IMPONIBILE = 720000
QUOZIENTE FAMILIARE = 720000
MONTANTE DELLE IMPOSTE = 394712
BREAK IN 230
```

Figura 6.17

6.5 INCIDENZA DI UNA REDDITO SUL POTERE D'ACQUISTO

Il potere di acquisto di una famiglia può essere definito dal reddito netto meno le spese fisse imposte (spese professionali, rimborsi di prestito per l'acquisto di una casa, imposte da pagare). Poichè in Italia le imposte sono pagate con un ritardo medio di un anno, si può considerare la loro incidenza sul potere d'acquisto leggermente più bassa poichè in questo periodo è possibile accumulare degli interessi, è sufficiente porre:

$$\frac{C}{1 + U} \qquad \text{con U è il tasso d'interesse}$$

per poter pagare con un anno di ritardo la somma C. Il tasso U dipende dall'utilizzatore.

Problema: Completare il programma precedente in modo che venga stampato il tasso dello scaglione più elevato per il quale il contribuente deve pagare delle imposte e poi che, a partire da un reddito addizionale netto (supposto senza spese) e da un tasso di interesse, calcoli l'aumento reale del potere di acquisto.

Per questo esercizio, si supporrà che il supplemento d'imposta è pagato in media un anno dopo aver incassato il reddito addizionale.

Si deve riflettere sulle condizioni di validità del programma e su altre modifiche per migliorare le condizioni di validità.

Soluzione: Il flow-chart generale del programma è molto "lineare", poichè non vi sono loop di calcolo e "calcoli iterativi" etc.

Poichè si farà due volte il calcolo dell'importo delle imposte, è consigliabile chiamare in questo caso una subroutine di calcolo.

Partendo dal programma precedente è sufficiente aggiungere una parte che legga il reddito addizionale, lo incorpori nel reddito precedente al fine di permettere un nuovo calcolo delle imposte dovute.In seguito, si calcoli l'interesse rappresentato dalla differenza tra i due importi che si suppone investiti durante l'anno, al tasso dato U dall'utilizzatore.

Si può infine calcolare il potere d'acquisto addizionale che è uguale a:

reddito supplementare — supplemento di imposta + interesse

Critica: In questo programma si suppone che il reddito addizionale non dia luogo a delle spese supplementari non deducibili (esempio: asilo d'infazia, riduzione dell'ammontare degli assegni familiari, etc.). Se è così, sarà sufficiente dedurre dal potere d'acquisto addizionale, dato dal programma, l'ammontare delle spese supplementari o la riduzione degli assegni, per ottenere l'aumento reale del potere d'acquisto.

```
10 REM  PROGRAMMA DI CALCOLO DEL MONTANTE DELL'IMPOSTA SUL REDDITO
20 REM  DELLE PERSONO FISICHE
30 REM  IL SOTTOPROGRAMMA LINEE 990-1170 LEGGE GLI SCAGLIONI E I TASSI
40 REM  E' NECESSARIO MODIFICARLO QUANDO CAMBIA L'ANNO
80 DIM F(15),T(15)
90 GOSUB 1000: PRINT "IMPOSTA SUL REDDITO DELL'ANNO ";A9: PRINT
100 INPUT "IMPOSTARE IL REDDITO NETTO ";R1
110 IF R1 < 0 THEN 100
120 INPUT "IMPOSTARE IL MONTANTE DELLA PENSIONE ";P
130 IF P < 0 THEN 120
140 INPUT "IMPOSTARE IL MONTANTE DELLE DEDUZIONI ";D
150 IF D < 0 THEN 140
160 R = .72 * R1 + P - D:R2 = R
170 INPUT "IMPOSTARE IL N. DEI COMPONENTI ";N
180 IF N < 0 OR N <  >  INT (N) THEN 170
190 Q = R / N
200 PRINT : PRINT "REDDITO IMPONIBILE = ";R
205 PRINT "QUOZIENTE FAMILIARE = ";Q
210 GOSUB 1200
220 PRINT "ENTRATE NELLO SCAGLIONE AL ";T1;" %"
230 PRINT "IL MONTANTE DELLE VS. IMPOSTE E' ";S
240 L = R1 + P - D - S:S1 = S
250 PRINT "DISPONETE DI ";L;" PER VIVERE"
260 PRINT
270 INPUT "QUAL'E' IL MONTANTE DI UN REDDITO ADDIZIONALE ";A
275 IF A <  = 0 THEN 500
280 R = R2 + .72 * A:Q = R / N
290 PRINT "NUOVO REDDITO IMPONIBILE = ";R
300 PRINT "NUOVO QUOZIENTE FAMILIARE = ";Q
310 GOSUB 1200
320 PRINT "NUOVO MONTANTE DELLE IMPOSTE = ";S
325 PRINT "A QUALE TASSO POTETE PIAZZARE LA SOMMA CHE RAPPRESENTA IL "
330 INPUT "SUPPLEMENTO DI IMPOSTA CHE PAGHERETE L'ANNO SUCCESS. ";U
335 L1 = A + S1 - S + 0.01 * U * (S - S1)
340 PRINT "DISPONETTE INFATTI DI UN POTERE DI ACQUISTO ADD. DI ";
345 PRINT L1
500 INPUT "ALTRO CASO DA TRATTARE ";R$
510 IF R$ <  > "SI" THEN  STOP
520 PRINT : GOTO 100
980 STOP

990 REM  SOTTOPROGRAMMA DI LETTURA DEGLI SCAGLIONI E DEI TASSI
1000 READ M,A9
1010 FOR I = 1 TO M
1020 READ F(I),A:T(I) = .01 * A
1030 NEXT I
1040 RETURN
1050 DATA  12,1978
1060 DATA  7925,5,8300,10
1080 DATA  9925,15,15700,20
1100 DATA  20625,25,25925,30
1120 DATA  31350,35,36175,40
1140 DATA  62600,45,86125,50
1160 DATA  105950,55,125050,60

1200 S = 0:T1 = 0: IF Q <  = F(1) THEN  RETURN
1210 FOR I = 2 TO M - 1
1220  IF Q < F(I) THEN 1270
1230  S = S + (F(I) - F(I - 1)) * T(I - 1)
1240  NEXT I
1250 S = S + (Q - F(M)) * T(M)
1260 GOTO 1280
1270 S = S + (Q - F(I - 1)) * T(I - 1)
1280 S =  INT (N * S + .49):T1 = 100 * T(I - 1)
1290 RETURN
```

Figura 6.18

Nota: In questo programma, la barra obliqua / separa più istruzioni sulla stessa linea.

Esempio di esecuzione:

```
IMPOSTA SUL REDDITO DELL'ANNO 1978
IMPOSTARE IL REDDITO NETTO 10000000
IMPOSTARE IL MONTANTE DELLA PENSIONE 500000
IMPOSTARE IL MONTANTE DELLE DEDUZIONI 180000
IMPOSTARE IL N. DEI COMPONENTI 4
REDDITO IMPONIBILE = 7520000
QUOZIENTE FAMILIARE = 1880000
ENTRATE NELLO SCAGLIONE AL 55 %
IL MONTANTE DELLE VS. IMPOSTE E' 4362850
DISPONETE DI 5957150 PER VIVERE
QUAL'E' IL MONTANTE DI UN REDDITO ADDIZIONALE 3000000
NUOVO REDDITO IMPONIBILE = 9680000
NUOVO QUOZIENTE FAMILIARE = 2420000
NUOVO MONTANTE DELLE IMPOSTE = 5658850
A QUALE TASSO POTETE PIAZZARE LA SOMMA CHE RAPPRESENTA IL
SUPPLEMENTO DI IMPOSTA CHE PAGHERETE L'ANNO SUCCESS. SI
?REENTER
SUPPLEMENTO DI IMPOSTA CHE PAGHERETE L'ANNO SUCCESS. 1650000
DISPONETTE INFATTI DI UN POTERE DI ACQUISTO ADD. DI 2.1385704E+10
ALTRO CASO DA TRATTARE 1210000
BREAK IN 510
```

Figura 6.19

CAPITOLO 7

GIOCHI

7.0 INTRODUZIONE

I giochi intellettuali che hanno luogo tra l'essere umano e il calcolatore interessano molte persone: i giovani per l'aspetto "gioco", i meno giovani per l'aspetto competizione contro questa macchina così rapida e gli specialisti che fanno ricerche sulle strategie.

L'esperienza mostra che la scrittura dei programmi di giochi costituisce, in generale, un lavoro lungo e difficile che non è, salvo casi eccezionali, alla portata del principiante. Questo è evidentemente il caso del gioco degli scacchi, ma per dei giochi più semplici si constata che il programma è piuttosto lungo e che le prestazioni del calcolatore non sono sempre buone.

Esistono dei giochi la cui programmazione è molto facile, o perchè non vi sono strategie da implementare (esempio il gioco alto-basso), o perchè la strategia è molto semplice (esempio il gioco di Marienbad).

Nel momento in cui le strategie si complicano il programma inevitabilmente si complica.

I tre giochi qui presentati sono particolarmente facili dal punto di vista della programmazione.

7.1 UN GIOCO: ALTO-BASSO

Prima parte: Bisogna trovare un numero intero N posto tra 0 e A, scelto a caso dal calcolatore. Quanto s'imposta un numero X, il calcolatore lo confronta con il numero da trovare N.

se $X = N$ il calcolatore stampa INDOVINATO DOPO I TENTATIVI, dove I rappresenta il numero dei tentativi effettuati dal giocatore,
se $X < N$ il calcolatore stampa E' ALTO
se $X > N$ il calcolatore stampa E' BASSO

Analisi: Per questo problema sarà necessario utilizzare 4 variabili:

A limite superiore dell'intervallo
N numero da trovare
X numero impostato dal giocatore
I numero di tentativi effettuati

La variabile A non è indispensabile, ma il suo utilizzo riduce il numero delle linee da modificare se si desidera cambiare il limite superiore dell'intervallo.

La variabile I è in effetti un "contatore" che si incrementa di una unità ad ogni nuovo tentativo.

Flow-chart: Tra tutti i metodi possibili, il flow-chart seguente sembra essere uno dei più semplici.

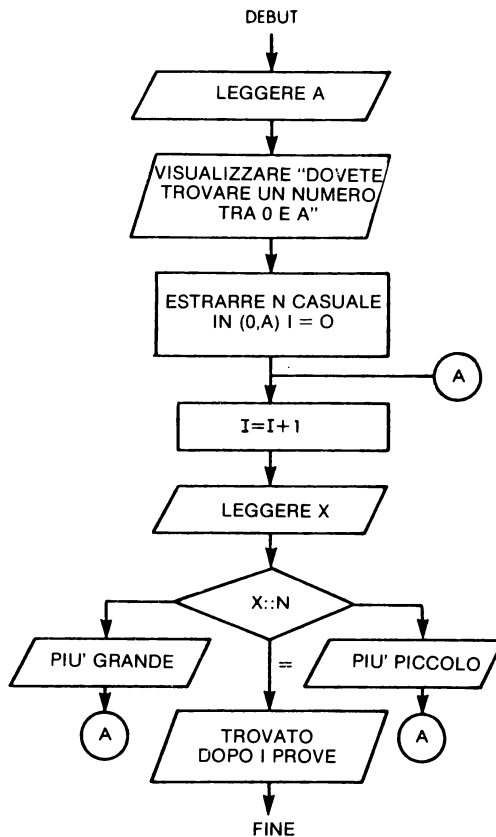



Figura 7.1

Per la programmazione, bisogna scegliere N intero casuale nell'intervallo (0, A). Per questo bisogna scrivere N = INT (A X RND(X). Questo metodo di scrittura può cambiare in minima parte da un sistema all'altro.

```
100  REM  CALCOLO DELLE MENSILITA' PER RIMBORSARE UN CAPITALE
110  INPUT "MONTANTE DEL CAPITALE PRESTATO: ";C
120  INPUT "TASSO ANNUALE IN %            : ";T
130  INPUT "NUMERO DELLE MENSILITA'       : ";N
140  T1 = (1 + T / 100) ^ (1 / 12) - 1
150  M = C * T1 / (1 - (1 + T1) ^ ( - N))
160  PRINT
170  PRINT "TASSO EQUIVALENTE = ";100 * T1;" % "
180  PRINT
190  PRINT "MONTANTE DI OGNI MENSILITA' = "; TAB( 35);M
200  PRINT
210  PRINT "SOMMA TOTALE RIMBORSATA     = "; TAB( 33);M * N
220  PRINT
230  INPUT "VOLETE IL DETTAGLIO DI ALCUNE MENSILITA' ? ";R$
240  IF R$ < > "SI" THEN  STOP
250  INPUT "NUMERI DELLE MENSILITA' ? ";W,B
260  PRINT : PRINT "NUMERO "; TAB( 8);" INTERESSE"; TAB( 23);"AMMORTAMENTO
     "
```

Figura 7.2

Seconda parte: Giocando ci si accorge che bisogna fare un piccolo sforzo per ricordare in quale intervallo si trova il numero, poichè, normalmente, questo intervallo diventa più piccolo. Per aiutare il giocatore, il calcolatore può ricordargli, dopo ogni tentativo errato, quali sono i limiti del nuovo intervallo.

Analisi: Per questo miglioramento saranno necessarie due nuove variabili C e D che rappresentano i limiti dell'intervallo.

Inizialmente C = 0 e D = A,

dopo un tentativo errato:

se $X < N$ porre C = MAX (C,X)
se $X > N$ porre D = MIN (D,X)

rigorosamente, se si ha la certezza che il giocatore non imposterà mai un numero superiore all'intervallo, sarà sufficiente porre C = X e D = X.

L'ordine di stampa diventa lo stesso in ogni caso errato poichè è sufficiente dare i limiti di intervallo.

Flow-chart: E' di immediata realizzazione sulla base del precedente.

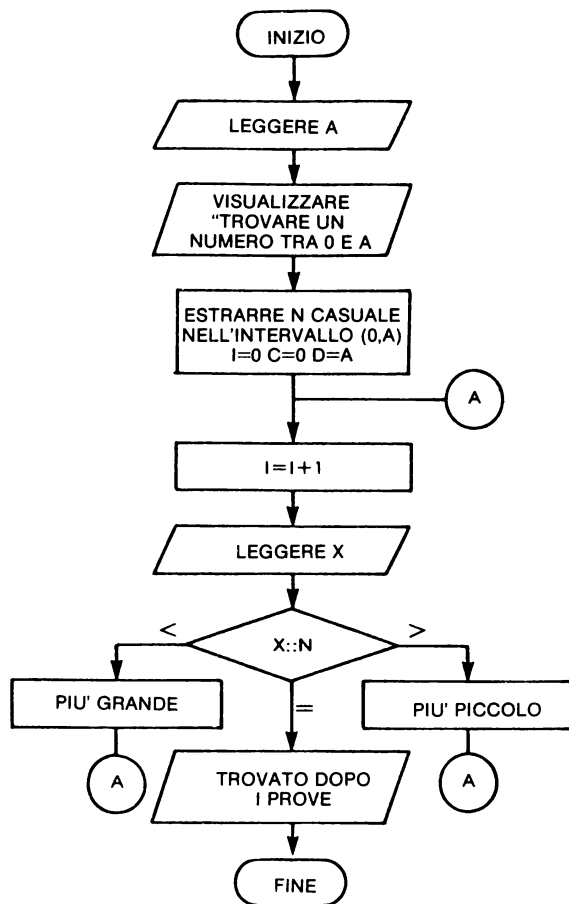



Figura 7.3

Terza parte: Un buon metodo per raggiungere rapidamente il numero cercato è quello di provare, a ogni tentativo, un numero che si trovi nel mezzo dell'intervallo.

Piuttosto che fare questo calcolo a mente è possibile modificare il programma per rendere automatica questa operazione.

Analisi: E' sufficiente nell'istruzione di stampa aggiungere il valore di (C + D)/2. Non vi è dunque alcuna modifica da apportare al flow-chart.

Programma:

```
10  INPUT "IMPOSTARE IL NUMERO MASSIMO ";A
20  PRINT
30  PRINT "DOVETE TROVARE UN NUMERO COMPRESO TRA 0 E ";A
40  PRINT
50  PRINT "IMPOSTARE IL TENTATIVO "
60  N=INT(A*RND(X))
70  I=0 : C=0 : D=A
80  I=I+1
90  INPUT X
100 IF X=N THEN GOTO 130
110 IF X<N THEN GOTO 112 ELSE GOTO 116
112 IF X>C THEN C=X : GOTO 118
116 IF X<D THEN D=X
118 PRINT "TRA ";C;" E ";D;" META' = ";(C+D)/2
120 GOTO 80
130 PRINT "INDOVINATO DOPO ";I;" TENTATIVI"
140 END
```

Figura 7.4

Quarta parte: Partendo dal momento in cui il giocatore introduce il numero suggerito dal programma, si può far giocare la macchina contro se stessa, in questo caso il giocatore diviene spettatore.

Analisi: Il flow-chart sarà di poco modificato, l'istruzione di lettura di X sarà rimpiazzata da X = INT((C + D) / 2) o da

$$X = INT((C + D + 1) / 2)$$

e l'istruzione che stampa i limiti dell'intervallo potrà essere soppressa. Invece se si desiderano conoscere i tentativi effettuati dalla macchina, sarà necessario inserire un'istruzione di stampa X.

Commento: In questo programma utilizzando il video la visualizzazione si effettua con una cadenza molto rapida. Per dare all'utilizzatore il tempo di leggere, è stata aggiunta l'istruzione SLEEP 5, che sospende l'esecuzione per 5 secondi dopo ogni gioco. Questa istruzione, SLEEP, non è disponibile per tutti i sistemi. Si può ottenere lo stesso risultato in altri modi:

— con l'istruzione WAIT quando disponibile,

— inserendo un loop di calcolo che impedisce all'unità centrale di passare da un tentativo all'altro troppo rapidamente.

```
10  INPUT "IMPOSTARE IL NUMERO MASSIMO ";A
20  PRINT
30  PRINT "DOVETE TROVARE UN NUMERO COMPRESO TRA 0 E ";A
40  PRINT
45  SLEEP 5
50  PRINT "IMPOSTARE IL TENTATIVO "
60  N=INT(A*RND(X))
70  I=0 : C=0 : D=A
80  I=I+1
90  X=INT((C+D)/2)
100 IF X=N THEN GOTO 130
110 IF X<N THEN GOTO 112 ELSE GOTO 116
112 IF X>C THEN C=X : GOTO 118
116 IF X<D THEN D=X
118 PRINT "TRA ";C;" E ";D;" META' = ";(C+D)/2
120 GOTO 80
130 PRINT "INDOVINATO DOPO ";I;" TENTATIVI"
135 GOTO 45
140 END
```

Figura 7.5

7.2 TROVA IL NUMERO

Questo gioco, che deriva da quello precedente, consiste nel trovare un numero scelto a caso dal computer dando una serie di intervalli (due numeri) che devono contenere il numero stesso. Il calcolatore dice:

— se il numero è stato inquadrato,
— se è più piccolo,
— se è più grande.

Per esempio se il numero da trovare fosse 55 e fossero proposti i numeri 18,24, il computer risponderebbe "PIU' GRANDE".

Primo problema: Studiare un semplice programma che permetta di giocare a questo gioco e che conti il numero dei tentativi effettuati dal giocatore.

Secondo problema: Proporre un programma più sofisticato che sia capace di capire se il giocatore gioca bene.

Soluzione al primo problema: Saranno necessarie le seguenti variabili:

A = numero massimo dell'intervallo (1,A),
N = numero da trovare,
X e Y limiti dell'intervallo proposti dal giocatore,
I contatore del numero dei tentativi

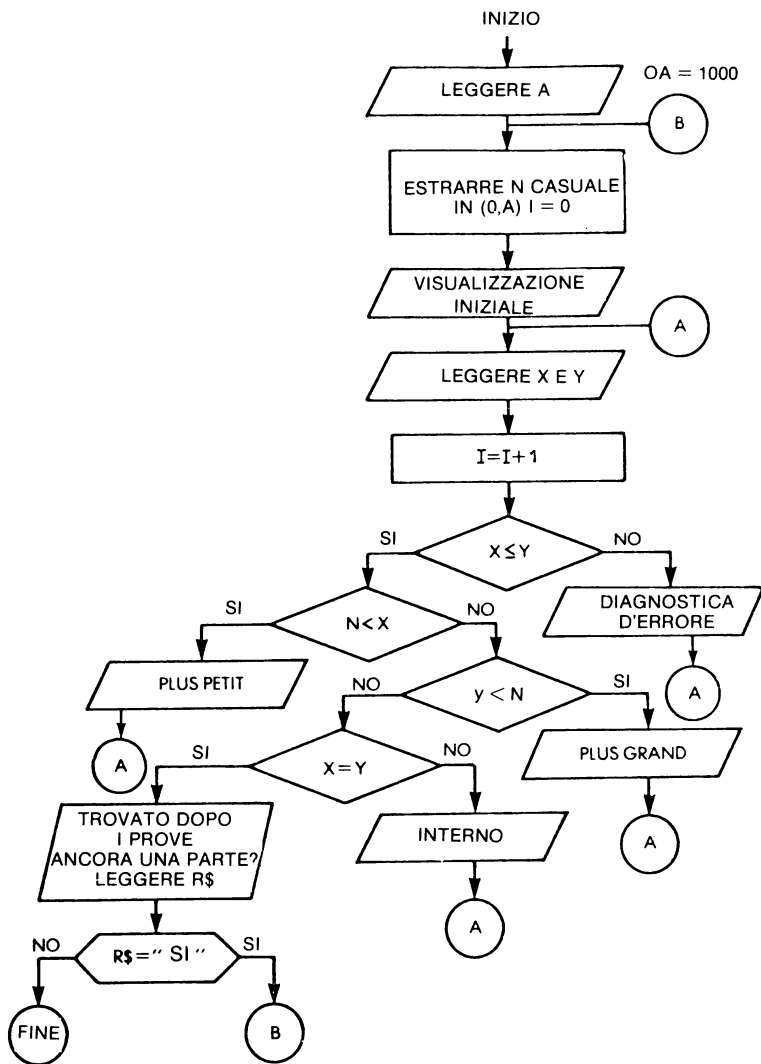



Figura 7.6

Il flow-chart di Fig.7.6 conduce facilmente al programma di Fig.7.7, come prima è possibile migliorare il programma in modo da poter suggerire al giocatore il migliore intervallo da provare e nello stesso tempo far giocare il calcolatore contro se stesso.

NOTA: Qui si ha l'interesse di dividere l'intervallo in tre intervalli uguali e di tentare l'intervallo centrale.

Esempio: per (0, 1000) tentare (333, 666)

Se la risposta è "Più grande" (667, 1000) provare allora l'intervallo (778, 889).

Questa strategia è la migliore poichè apporta il massimo dell'"informazione", la parola informazione è qui utilizzata in senso matematico e non informatico.

```
100 REM  TROVA IL NUMERO
110 A = 1000
115 N =  INT (A *  RND (1))
120 I = 0
130 PRINT "TROVARE IL NUMERO COMPRESO TRA 0 E ";A;" PER INQUADRAMENTO"
140 INPUT X,Y
145 I = I + 1
150 IF X <  = Y THEN 180
160 PRINT "IL PRIMO NUMERO NON DEV'ESSERE PIU' GRANDE DEL SECONDO"
170 GOTO 140
180 IF N < X THEN 210
190 IF N > Y THEN 220
200 IF X = Y THEN 230
205 PRINT "CENTRATO ": GOTO 140
210 PRINT "PIU' PICCOLO ": GOTO 140
220 PRINT "PIU' GRANDE ": GOTO 140
230 PRINT "TROVATO DOPO ";I;" TENTATIVI ": PRINT
240 INPUT "ANCORA UNA PARTITA ";R$
250 IF R$ = "SI" THEN 115
260 END
```

Figura 7.7

Esempio di esecuzione:

```
TROVARE IL NUMERO COMPRESO TRA 0 E 1000 PER INQUADRAMENTO
?400,600
PIU' PICCOLO
?200,400
CENTRATO
?300,350
PIU' PICCOLO
?250,300
CENTRATO
?270,300
CENTRATO
?290,300
PIU' PICCOLO
?280,290
CENTRATO
?285,1
CENTRATC
?285,285
PIU' GRANDE
?287,287
PIU' GRANDE
?2878,288
IL PRIMO NUMERO NON DEV'ESSERE PIU' GRANDE DEL SECONDO
?288,288
PIU' GRANDE
?289,289
TROVATO DOPO 13 TENTATIVI
ANCORA UNA PARTITA
```

Figura 7.8

7.3 GIOCO DI MARIENBAD

Questo gioco molto semplice ha la particolarità di offrire una "strategia" vincente per il giocatore che muove per secondo. Le regole del gioco sono le seguenti: Si dispongono all'inizio del gioco 21 fiammiferi che potranno essere rimossi da ciascun giocatore, al proprio turno, nella misura da 1 a 4. Il giocatore costretto a prendere l'ultimo fiammifero perde.

La strategia del giocatore che gioca per secondo è di completare a 5 il numero dei fiammiferi presi dal primo giocatore. Così il primo giocatore si troverà di fronte a gruppi composti rispettivamente da 21, 16, 11, 6 e 1 fiammiferi e sarà quindi obbligato a prendere l'ultimo.

Esempio:

Giocatore 1	Giocatore 2	Resto
3	2	16
2	3	11
4	1	6
1	4	1
1 e il primo giocatore perde		

Problema: Costruire un programma che dia al computer il ruolo del secondo giocatore applicando quindi la strategia vincente.

Il programma dovrà cercare eventuali "imbrogli" del primo giocatore.

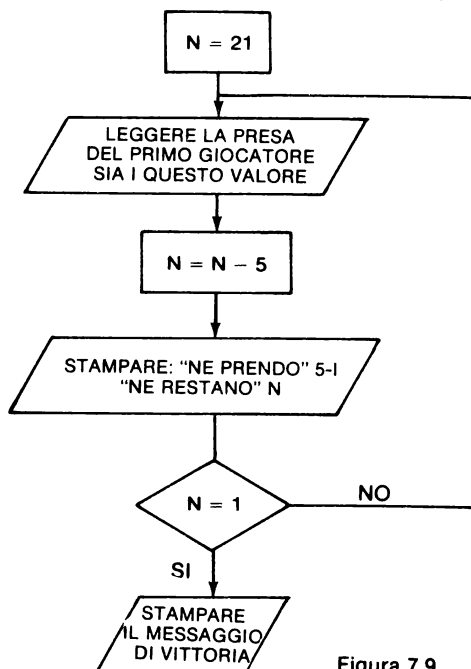



Figura 7.9

Soluzione: l'algoritmo può essere rappresentato con il flow chart di principio 7.9.

Per conteggiare gli eventuali "imbrogli" del primo giocatore, si aggiungeranno dei test.

— test che I è intero
— test che I è compreso tra 1 e 4

Se un test fornisce una risposta negativa, sarà necessario visualizzare un nuovo messaggio di errore e ritornare alla lettura del valore di I per poter reimpostare la giocata.

```
100  REM  GIOCO DI MARIENBAD
110  REM
160  PRINT "CI SONO 21 FIAMMIFERI. CHI PRENDE"
170  PRINT "L'ULTIMO PERDE. SI GIOCA A TURNO "
175  PRINT "E SI DEVONO PRENDERE 1,2,3 O 4 "
180  PRINT "FIAMMIFERI ALLA VOLTA."
185  PRINT :N = 21
190  INPUT "QUANTI NE PRENDI ? ";I
200  IF I <  >  INT (I) THEN 400
210  IF I < 1 OR I > 4 THEN 430
220  N = N - 5
230  PRINT "NE PRENDO ";5 - I;" NE RESTANO ";N
240  IF N > 1 THEN 190
250  PRINT "DEVI PRENDERE L'ULTIMO "
260  PRINT
270  PRINT "HO VINTO !!": PRINT
280  INPUT "ANCORA UNA PARTITA ? ";R$
290  IF R$ = "SI" THEN 185
300  STOP
400  PRINT "IL NUMERO DEV'ESSERE INTERO "
410  GOTO 190
430  PRINT "NON IMBROGLIARE !!, DEVI PRENDERE 1,2,3 O 4 FIAMMIFERI"
440  GOTO 190
```

Figura 7.10

Esempio di esecuzione:

```
CI SONO 21 FIAMMIFERI. CHI PRENDE
L'ULTIMO PERDE. SI GIOCA A TURNO
E SI DEVONO PRENDERE 1,2,3 O 4
FIAMMIFERI ALLA VOLTA.
QUANTI NE PRENDI ? 1
NE PRENDO 4 NE RESTANO 16
QUANTI NE PRENDI ? 1
NE PRENDO 4 NE RESTANO 11
QUANTI NE PRENDI ? 2
NE PRENDO 3 NE RESTANO 6
QUANTI NE PRENDI ? 4
NE PRENDO 1 NE RESTANO 1
DEVI PRENDERE L'ULTIMO
HO VINTO !!
ANCORA UNA PARTITA ? SI
QUANTI NE PRENDI ? 3
NE PRENDO 2 NE RESTANO 16
QUANTI NE PRENDI ? 2
NE PRENDO 3 NE RESTANO 11
QUANTI NE PRENDI ? 3
NE PRENDO 2 NE RESTANO 6
QUANTI NE PRENDI ? 1
NE PRENDO 4 NE RESTANO 1
DEVI PRENDERE L'ULTIMO
HO VINTO !!
ANCORA UNA PARTITA ? NO
BREAK IN 300
```

Figura 7.11

Il programma di Fig.7.10 scritto partendo dal flow-chart 7.9 comporta i due test alle linee 200 e 210, i messaggi di errore alle linee 400 e 430 che poi rinviano alla lettura dell'input.

7.4 CONCLUSIONI

I tre giochi qui presentati sono molto semplici da programmare per due ragioni:

— assenza di strategia o strategia molto elementare,
— nessuna "posizione strategica da valutare" come nei giochi tipo dama o scacchi.

Per tutti i giochi che si svolgono su una "scacchiera" (OTELLO, DAMA, SCACCHI), si devono manipolare degli indici che aggiungono una complicazione supplementare alla difficoltà della strategia.

Si consiglia ai lettori interessati a questo tipo di problema, di affrontare i giochi gradatamente più complessi, per esempio seguendo la seguente progressione:

Gioco del NIM (simile al gioco di Marienbad ma con più gruppi)
Gioco del MASTER MIND.
Gioco di OTELLO implementando prima una strategia semplice poi provare a raffinarla progressivamente.

CAPITOLO 8

RICERCA DEL METODO OPERATIVO

8.0 INTRODUZIONE

Una parte importante dei problemi di ricerca operativa porta al trattamento di grafici (Esempio: Problema del rappresentante, PERT, ricerca topologica etc.). Per questa categoria di problemi, la manipolazione degli indici è molto delicata, anche per un programmatore esperto.

Inoltre, gli indici hanno, in generale, dei valori interi e sono molto più adatti per questo tipo di problemi gli interpreti BASIC che accettano la rappresentazione intera e la rappresentazione in "virgola flottante".

Il lettore è quindi invitato a non affrontare questa parte fino a quando non abbia acquisito una sufficiente pratica con gli esercizi precedenti.

8.1 RICERCA TOPOLOGICA

Siano T1, T2, ..., T_N lavori da realizzare rispettando dei vincoli di precedenza: un certo numero di coppie (*i, j*) indicano che il lavoro *j* non può essere intrapreso fino a quando non è stato terminato il lavoro i; un' ultima coppia (*o, o*) indica che la lista è terminata.

Problema: Partendo dai seguenti dati - numero dei lavori e liste di coppie (*i, j*) - trovare un ordine di esecuzione dei differenti lavori soddisfacendo ai vincoli di precedenza.

Per risolvere questo problema si potrà utilizzare il seguente metodo:

— una tabella T inizializzata a 0 viene modificata dalla lettura delle coppie (I, J), quando si legge una costrizione (*i, j*) che significa che il lavoro J non può essere intrapreso fino quando il lavoro i non è terminato, si pone la costante 1 in T (I, J),

— in seguito si cerca un lavoro eseguibile che non abbia costrizioni, o le cui costrizioni siano state soddisfatte. Un lavoro K è caratterizzato da:

T(I, K) = 0 qualunque sia I

L'esecuzione di questo lavoro K libera alcuni vincoli. Si esegue dunque:

$$T(K, J) = 0 \text{ qualunque sia } J$$

Questo lavoro può essere scelto e il suo numero può essere stampato. Per non doverlo selezionare un'altra volta si eporrà:

$$T(K, K) = 1$$

Si continua cercando un nuovo lavoro fino a che non si è terminato.

Due casi si possono allora presentare:

Primo caso: Si contano i lavori selezionati ed è stato possibile stampare gli N lavori. Si è dunque trovata la soluzione.

Secondo caso: Dopo aver fatto tutte le prove possibili, si constata che il numero di lavori stampati è inferiore a N. Il problema non ammette quindi soluzione, è necessario dunque stampare un messaggio.

Si applicherà il programma all'esempio dato dallo schema di Fig.8.1 nel quale, una freccia che và per esempio dal nodo 8 verso il nodo 5, significa che il lavoro 5 non può essere intrapreso prima che il lavoro 8 sia stato ultimato.

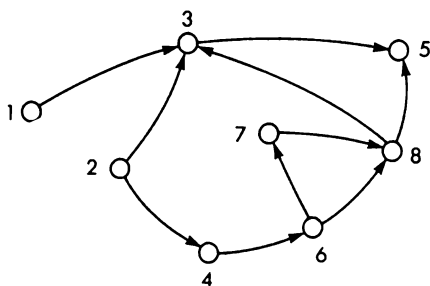



Figura 8.1

Questo esempio si concretizzerà con i dati della Fig.8.2 che segue:

```
1000  DATA  8
1010  DATA  1,3
1020  DATA  2,3
1030  DATA  8,3
1040  DATA  3,5
1050  DATA  4,6
1060  DATA  6,7
1070  DATA  6,8
1080  DATA  2,4
1090  DATA  8,5
1100  DATA  7,8
1110  DATA  0,0
```

Figura 8.2

Soluzione: Si scomporrà il problema in tre parti come suggerisce il seguente metodo:

— inizializzazione della tabella a 0,
— lettura dei dati e preparazione della tabella T,
— esecuzione dell'algoritmo.

Nel programma dato in Fig.8.3, queste 3 parti sono l'oggetto di tre sottoprogrammi.

Paragrafo inizializzazione a 0: per alcuni interpreti, tutte le variabili sono, all'inizio, inizializzate a 0, ma poichè questo non è prerogativa di tutti i sistemi, è preferibile farlo comunque come alle linee 500-540.

Lettura di dati e preparazione della tabella T e stampa dei dati: durante questa lettura, si verifica che i numeri dei lavori sono verosimili e che un vincolo è caratterizzato da numeri differenti (altrimenti un lavoro dovrà essere preceduto da se stesso). Se si trova un errore, viene stampato un messaggio. Se un vincolo è accettato, si pone: T (K, L) = 1.

Esecuzione di algoritmo (linee 800-960):
questa parte è più corta di quella precedente, poichè è semplice e vi è una sola istruzione di uscita.

Si noti che per un grafico che comporta un totale di *n* lavori, vi sono al massimo $\frac{n(n-1)}{2}$ vincoli. Questo appare nell'istruzione FOR della linea 605.

```
100 REM  QUESTO PROGRAMMA DI RICERCA TOPOLOGICA DETERMINA IN QUALE ORDINE
105 REM  SI POSSONO ESEGUIRE DEI LAVORI CHE DEVONO SODDISFARE DELLE
110 REM  COSTRIZIONI DI PRECEDENZA.
120 PRINT "RICERCA TOPOLOGICA ": PRINT
150 DIM T(20,20):N9 = 20
170 REM  INIZIALIZZAZIONE DELLA TABELLA T
180 GOSUB 500
190 REM  LETTURA, CONTROLLO E STAMPA DEI DATI
200 GOSUB 600
210 REM  CHIAMATA DELL'ALGORITMO
220 GOSUB 800
230 STOP
500 FOR I = 1 TO N9
510  FOR J = 1 TO N9
520   T(I,J) = 0
530   NEXT J
540  NEXT I: RETURN
600 READ N: PRINT "NUMERO DEI LAVORI = ";N: PRINT
603 PRINT "LISTA DELLE COSTRIZIONI ": PRINT
605 FOR I = 1 TO (N - 1) * (N - 1)
610  READ K,L
620  IF K = 0 AND L = 0 THEN 720
630  IF K <  > L THEN 650
640   PRINT "ERRORE POICHE' 2 LAVORI HANNO LO STESSO NUMERO ";K: STOP
650  IF K > 0 AND K < N9 THEN 670
660   PRINT "ERRORE DI NUMERAZIONE DEL LAVORO ";K: STOP
670  IF L > 0 AND L < N9 THEN 690
680   PRINT "ERRORE DI NUMERAZIONE DEL LAVORO ";L: STOP
690  T(K,L) = 1: PRINT K; TAB( 6);L
700  NEXT I
710 PRINT "ERRORE NEI DATI ": STOP
720 C = I - 1: PRINT
730 PRINT "NUMERO DELLE COSTRIZIONI = ";C: RETURN
800 PRINT : PRINT "ORDINE DEI LAVORI ":I = 0
810 K = 1
820  FOR J = 1 TO N
830   IF T(J,K) = 1 THEN 920
840   NEXT J
850  I = I + 1: PRINT  TAB( 5 * I - 4);K;
860  FOR J = 1 TO N
870   T(K,J) = 0
880   NEXT J
890 T(K,K) = 1: GOTO 810
920 K = K + 1: IF K <  = N THEN 820
930 IF I = N THEN  RETURN
940 PRINT "NESSUNA SOLUZIONE POICHE' NON SI PUO' REALIZZARE ";N - I;" LAV
    ORI "
960 STOP
```

Figura 8.3

```
RICERCA TOPOLOGICA

NUMERO DEI LAVORI = 8

LISTA DELLE COSTRIZIONI

1       3
2       3
8       3
3       5
4       6
6       7
6       8
2       4
8       5
7       8

NUMERO DELLE COSTRIZIONI = 10

ORDINE DEI LAVORI
1  2  4  6  7  8  3  5
```

Figura 8.4

8.2 IL PERCORSO PIU' BREVE IN UN GRAFICO

Il programma qui proposto tratta il caso dell'ordinamento in sequenza di lavori di durata conosciuta.

La numerazione dei lavori è stata stabilita per mezzo di numeri crescenti al fine di semplificare la programmazione, ciò permette così di ottenere dei buoni risultati anche con un personal computer.

Tenuto conto della lunghezza del programma, non verrà posto un problema da risolvere ma verrà presentata direttamente una soluzione.

Presentazione dei dati:

L'insieme dei lavori può essere rappresentato da un grafico orientato avente un'entrata ed in principio una sola uscita. Questo grafico orientato non deve comportare dei "cicli". Per esempio si potrà avere il grafico della Fig.8.5.

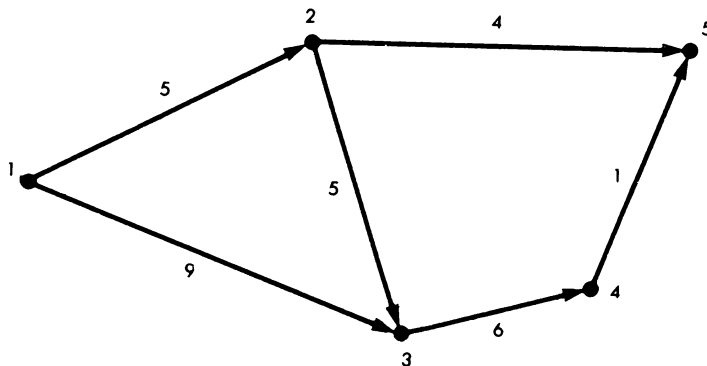



Figura 8.5

Nel grafico ogni freccia corrisponde ad un lavoro. Per esempio tra i nodi 2 e 5 figura un lavoro la cui durata è 4.

Infatti ogni lavoro è caratterizzato da:

— un numero di nodo di partenza,
— un numero di nodo di arrivo,
— una durata (l'unità può essere una qualsiasi),
— una designazione.

Così il grafico della Fig.8.5 corrisponde al listato dei lavori riportato in Fig.8.6.

```
2010  DATA  1,2,5,"SOLLEVAMENTO"
2020  DATA  1,3,9,"SMONTAGGIO"
2030  DATA   2,3,5,"CAMBIO RUOTE"
2040  DATA  2,5,4,"REVISIONE"
2050  DATA  3,4,6,"DISCESA"
2060  DATA  4,5,1,"SERRAGGIO"
2070  DATA  0,0,0,"
]
```

Figura 8.6

Per poter leggere questi dati, si utilizzeranno tre tabelle:

— tabella S che conterrà i nodi di partenza di ogni lavoro
— tabella F che conterrà i nodi di arrivo di ogni lavoro
— tabella D che conterrà la durata di ogni lavoro.

Una stringa di caratteri C$ conterrà le designazioni. Il programma è stato limitato al trattamento di problemi con non più di 20 lavori. Per semplificare la lettura una variabile, stringa di caratteri, D$ limitata a 20 caratteri riceverà la designazione di ogni lavoro.

Tutte le tabelle saranno quindi dimensionate a 20. La stringa C$ sarà una lunghezza di 20 x 20 = 400 caratteri.

Un sottoprogramma di lettura effettua il seguente lavoro:

— lettura dei dati,
— individuazione della fine dei dati caratterizzati da N1 = N2 = 0,
— controllo N1 < N2,
— inizializzazione di alcune tabelle,
— conteggio del numero dei lavori. Il risultato è posto nella variabile N.

```
950  REM  SOTTOPROGRAMMA DI LETTURA E STAMPA DEI DATI
980  PRINT "DA  A     DURATA DESIGNAZIONE"
990  PRINT
1000  FOR I = 1 TO N9
1010   READ N1,N2,D(I),D$
1020   IF N1 = 0 AND N2 = 0 THEN 1100
1025   IF N1 < N2 THEN 1050
1030    PRINT "NUMERO NON INCROCIANTE": STOP
1050 E(N1) = 0:E(N2) = 0:L(N1) = 0:L(N2) = 0
1060 S(I) = N1:F(I) = N2:C$(20 * I - 19) = D$
1070  PRINT S(I); TAB( 5);F(I); TAB( 13);D(I); TAB( 18);D$
1080  NEXT I
1090  N = N9: GOTO 1110
1100  N = I - 1
1110 PRINT
1120 PRINT "NUMERO DEI LAVORI = ";N
1130 RETURN
```

Figura 8.7

In questo sottoprogramma dato dal listato di Fig.8.7, le variabili avranno il seguente significato:

N9 variabile che è stata inizializzata dal valore 20 al lancio del programma principale.

N1 e N2 variabili che ricevono al momento della lettura rispettivamente il numero del nodo di partenza ed il numero del nodo di arrivo, i valori di N1 e N2 vengono trasferiti nelle tabelle S e F.

Tabella E: tempo minimo di inizio di ogni lavoro.
E è inizializzato a 0.

Tabella L: tempo massimo di esecuzione di ogni lavoro volendo mantenere la tempistica stabilita dal progetto.
L è inizializzato a 0.

Il programma principale, dopo aver fatto leggere i dati dal sottoprogramma già esaminato, procede con il calcolo del tempo minimo di inizio di ogni lavoro.

Si prenda l'esempio del grafico in Fig.8.5. Il lavoro che va dal nodo 3 al nodo 4, non può cominciare fino a che i lavori che terminano in 3 non siano stati ultimati. Questo tempo E(3) è dunque caratterizzato da:

$$\left.\begin{array}{l} 9 \leqslant E(3) \\ 5 + 5 \leqslant E(3) \end{array}\right\} \text{ questo dà } E(3)=10$$

In generale si potrà partire da E(1)=0 e fare il seguente calcolo:

$$E(F(I)) = MAX\,[E(F(I)),\ E(S(I)) + D(I)]$$

questo si trova alle linee 270-300.

Per calcolare il tempo massimo di realizzazione di ogni lavoro si partirà in senso inverso, per l'ultimo nodo si avrà:

$$L (F(N)) = E (F(N))$$

L rappresenta la tabella dei tempi massimi di esecuzione, così:

$$L(S(I)) = MIN [L(S(I)), L(F(I)) - D(I)]$$

questo calcolo è realizzato alle linee 320 - 360.
Partendo dal momento in cui si conosce per ogni lavoro:

— il tempo minimo di inizio E(S(I)),
— il tempo massimo di esecuzione L(F(I)),
— la durata normale di esecuzione D(I).

Si ottiene la durata del ritardo massimo che si può ammettere su questo lavoro con:

$$F1(I) = L(F(I)) — E(S(I) — D(I))$$

se F1(I) = 0 significa che ogni ritardo nell'esecuzione di un lavoro ritarderà l'intero progetto. La variabile C1 è destinata a contare questi lavori che mostrano tempi critici. Questo calcolo è realizzato nelle linee 410-440.

E' possibile ora stampare, per ogni lavoro, le seguenti informazioni:

— numero del nodo di partenza — numero del nodo d'arrivo — durata	dati
— tempo minimo di inizio — tempo massimo di esecuzione — durata del possibile ritardo	risultati del calcolo

Questo è realizzato dalla sequenza di istruzioni 495-550.

In seguito vi è il calcolo della durata totale del cammino critico per mezzo della variabile C3.

$$C_3 = MAX (C3,L(F(I)))$$

linee 570 - 590.

Poichè il cammino critico comporta dei lavori la cui durata di ritardo ammissibile è zero, lo si può stampare partendo dal nodo d'inizio (linee 595-720)

— ricerca del lavoro di partenza (linee 640-660),
— stampa di questo lavoro,

— ricerca del lavoro successivo (linee 700 - 720), stampa e proseguimento fino alla fine.

Il flow-chart di questa parte è mostrato in Fig.8.8.

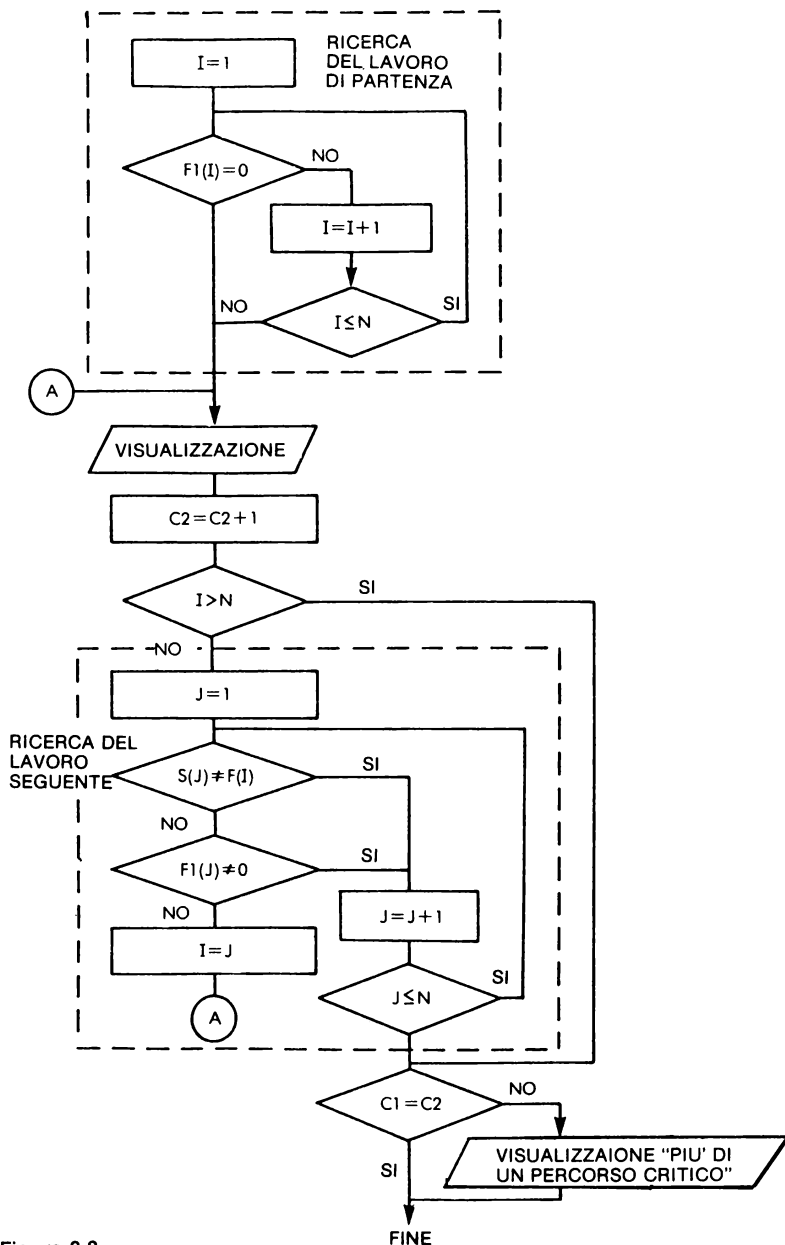



Figura 8.8

```
100 REM  IL PERCORSO PIU' BREVE CON UN GRAFICO
110 DIM S(20),F(20),D(20),E(20),L(20),F1(20),C$(400),D$(20)
115 N9 = 20
120 REM
130 REM  LETTURA E STAMPA DEI DATI
150 GOSUB 980
250 REM  INIZ. E CALCOLO DELLA DATA D'INIZIO LAVORI
260 C1 = 0:C2 = 0:C3 = 0
270 FOR I = 1 TO N
280  M1 = E(S(I)) + D(I)
290  IF E(F(I)) <  = M1 THEN E(F(I)) = M1
300  NEXT I
310 REM
320 L(F(N)) = E(F(N))
330 FOR I = N TO 1 STEP  - 1
340  L1 = S(I):M2 = L(F(I)) - D(I)
350  IF L(L1) >  = M2 OR L(L1) = 0 THEN L(L1) = M2
360  NEXT I
400 REM
410 FOR I = 1 TO N
420  F1(I) = L(F(I)) - E(S(I)) - D(I)
430  IF F1(I) = 0 THEN C1 = C1 + 1
440  NEXT I
495 PRINT
500 PRINT "ANALISI DEL CAMMINO CRITICO"
510 PRINT
520 PRINT "DA  ";"A","MIN.INIZ.","MAX.ESEC.","NORM.DUR."
525 PRINT
530 FOR I = 1 TO N
535  PRINT S(I)"  ";F(I).E(S(I)),L(F(I)),F1(I),C$(20 * I - 19)
550  NEXT I
560 REM
570 FOR I = 1 TO N
580  IF L(F(I)) > C3 THEN C3 = L(F(I))
590  NEXT I
595 PRINT
600 PRINT "LUNGHEZZA DEL CAMMINO CRITICO ";C3
610 PRINT
620 PRINT "VA DA","A"
630 PRINT
640 FOR I = 1 TO N
650  IF F1(I) = 0 THEN 670
660  NEXT I
670 PRINT S(I),F(I)
680 C2 = C2 + 1
690 IF I > N THEN 730
700 FOR J = 1 TO N
710  IF S(J) <  > F(I) OR F1(J) <  > 0 THEN 720
715   I = J: GOTO 670
720  NEXT J
730 IF C1 <  > C2 THEN  PRINT "PIU' DI UN CAMMINO CRITICO"
740 PRINT
800 STOP
950 REM  SOTTOPROGRAMMA DI LETTURA E STAMPA DEI DATI
980 PRINT "DA","A","DURATA","DESIGNAZ."
990 PRINT
1000 FOR I = 1 TO N9
1010 READ N1,N2,D(I),D$
1020 IF N1 = 0 AND N2 = 0 THEN 1100
1025 IF N1 < N2 THEN 1050
1030 PRINT "NUMERO NON INCROCIANTE": STOP
1050 E(N1) = 0:E(N2) = 0:L(N1) = 0:L(N2) = 0
1060 S(I) = N1:F(I) = N2:C$(20 * I - 19) = D$
1070 PRINT S(I),F(I),D(I),D$
1080 NEXT I
1090 N = N9: GOTO 1110
1100 N = I - 1
```

Figura 8.9

L'esecuzione porta alla figura seguente costituita da tre parti:

```
DA          A           DURATA      DESIGNAZ.

1           2           5           SOLLEVAMENTO
1           3           9           SMONTAGGIO
2           3           5           CAMBIO RUOTE
2           5           4           REVISIONE
3           4           6           DISCESA
4           5           1           SERRAGGIO

NUMERO DEI LAVORI = 6

ANALISI DEL CAMMINO CRITICO

DA  A       MIN.INIZ.   MAX.ESEC.   NORM.DUR.

1  2        0           5           0           SOLLEVAMENTO
1  3        0           10          1           SMONTAGGIO
2  3        5           10          0           CAMBIO RUOTE
2  5        5           17          8           REVISIONE
3  4        10          16          0           DISCESA
4  5        16          17          0           SERRAGGIO

LUNGHEZZA DEL CAMMINO CRITICO 17

VA DA       A

1           2
2           3
3           4
4           5


BREAK IN 800
```

Figura 8.10

— stampa dei dati e numero totale dei lavori,
— analisi del cammino critico, per ogni lavoro:

  — numero del nodo di partenza,
  — numero del nodo di arrivo,
  — data di inizio lavoro,
  — data di fine per non ritardare la realizzazione del progetto
  — durata delle soste,

— itinerario del cammino critico.

8.3 IL PROBLEMA DEL COMMESSO VIAGGIATORE

Un rappresentante deve visitare i suoi clienti disseminati in N città. Si conoscono i costi d_{ij} per andare dalla città *i* alla città *j*. I costi possono essere espressi in distanza chilometrica o con altre unità.

In quale ordine il rappresentante deve visitare i suoi clienti per minimizzare il

costo dello spostamento ? In questo problema il rappresentante dovrà, dopo il suo giro tornare al punto di partenza.

Esempio:

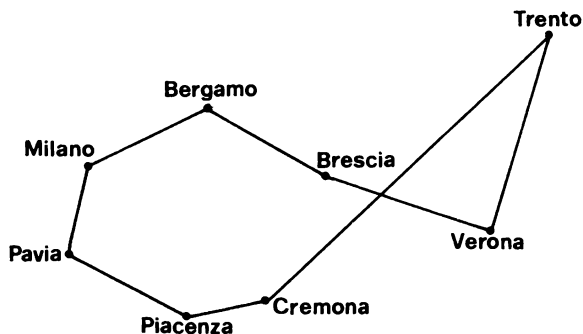



Figura 8.11

Metodo suggerito: Non si utilizzerà l'algoritmo esatto che è abbastanza complesso e lento nell'esecuzione. Si utilizzerà il metodo euristico seguente:

1. Si sceglie una città di partenza.
2. Si decide di andare poi nella città più vicina.
3. Quindi si andrà nell'altra città più vicina non ancora visitata, e così di seguito fino a che non si siano visitate tutte le città. Si ritorna poi alla città di partenza. Si noti il costo del percorso così determinato e si ricominci scegliendo tutte le città come punto di partenza.

Problema:

1. Analizzare il problema in vista di una scomposizione in parti relativamente corte.
2. Stabilire un flow-chart di principio molto coinciso facendo apparire dei sottoprogrammi.
3. Stabilire in seguito dei flow-chart di dettaglio per ciascun sottoprogramma.
4. Scrivere il programma.

Si utilizzeranno le seguenti variabili:

V$ Matrice di stringhe di caratteri che conterrà il nome delle città.

D Tabella a due dimensioni che contiene la matrice dei costi

d_{ij} = costo per andare dalla città i alla città j

$d_{ii} = 0$

T Tabella che contiene una soluzione in corso di esecuzione.

T2 Tabella che contiene la migliore soluzione trovata durante l'esecuzione dell'algoritmo.

S Costo della migliore soluzione trovata.

C Costo della soluzione in corso di studio.

Soluzione: Questo problema non è difficile a condizione di lavorare con metodo.

1. *Analisi*: Il programma totale comporterà differenti parti:

— lettura dei dati,
— stampa dei dati,
— ricerca del migliore itinerario (parte algoritmo),
— stampa di quell'itinerario.

Per facilitare la messa a punto della parte relativa all'algoritmo è possibile la stampa di itinerari provvisori. E' quindi facile realizzare la visualizzazione per mezzo di un sottoprogramma. Sarà sufficiente quindi inserire un'istruzione GOSUB per questa messa a punto.

La matrice dei costi può essere simmetrica o asimmetrica. Si esamineranno nella lettura i due casi. Così, il lavoro dell'utilizzatore sarà ridotto se c'è una simmetria.

Si giunge così al flow-chart di principio di Fig.8.12.

La visualizzazione della matrice dei costi potrà essere la stessa se è simmetrica oppure no. Per contro si dovrà tener conto della lunghezza delle linee.

Per ottenere una corretta stampa si potrà utilizzare una tabella V$ di stringhe di caratteri che conterrà il nome delle città. La matrice dei costi sarà rappresentata da una tabella quadrata D a due dimensioni.

Nel flow-chart di Fig.8.12, si è evitato di descrivere l'algoritmo. Progressivamente entreremo nel dettaglio; occorrerà costruire una descrizione, memorizzarla e comparare il suo costo con un' altra soluzione. Si utilizzeranno le variabili che seguono.

Una tabella T che conterrà la sequenza dei numeri delle città nell'ordine in cui appaiono nel circuito. Si avrà dunque:

$$C = \sum_{I=1}^{N-1} D(T(I), T(I+1)) + D(T(N), T(1))$$

Si potrà ora costruire il flow-chart di Fig.8.13, nel quale la ricerca della città successiva non è stata dettagliata.

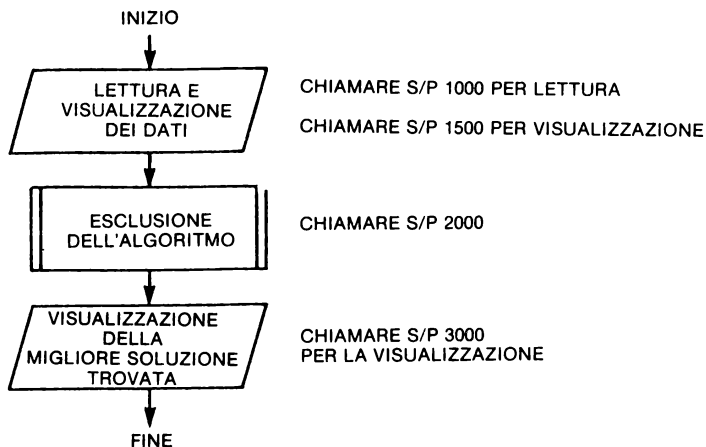



Figura 8.12

Per questo si esamina la situazione in corso di definizione del percorso: si supponga che L—1 città siano state trovate, i loro numeri sono in T(1) a T(L—1). Successivamente si cercherà tra tutte quelle città il cui numero J soddisfa:

$$J \neq T(K) \text{ per } K = 1, 2, \dots , L - 1$$

quelle di cui il costo D(T(L—1),J) è minimo al fine di selezionare e immagazzinare in T(L) i loro numeri. Questo porta al flow-chart di Fig.8.14 che permette di intraprendere la programmazione.

Il programma è stato scritto per un microcalcolatore che ammette le tabelle di stringhe di caratteri. Per un microcalcolatore che non ammette questo tipo di tabella, occorrerà apportare delle modifiche simili a quelle che figurano nel programma PERT (vedere problema 9.2).

Per facilitare la comprensione del programma, questo è stato diviso in sottoprogrammi, il programma principale provvede alla chiamata di quattro subroutine.

995 lettura della tabella dei costi
1500 stampa della tabella dei costi
2000 algoritmo
3000 stampa della soluzione trovata

INIZIO
S = 1E 38
I = 1
C = 0
T(1) = I
L = 2
RICERCA DELLA
CITTA' SEGUENTE
SIA L1 N° CITTA'
TROVATA,
CORSO C1
MEMORIZZAZIONE
T(L) = L1
CUMOLO C = C + C1
L = L + 1
L ≤ N
SI
NO
C = C + D(T(N), T(1))
creazione di un percorso partendo dal nodo I
MEMORIZZAZIONE
DI QUESTA
SOLUZIONE
SE È LA MIGLIORE
I = I + 1
I ≤ N
SI
NO
RETURN
PER RITORNARE AL PUNTO
DI PARTENZA

Figura 8.13

INIZIO
C1 = 1E 38
J = 1
K = 1
LA CITTA' J È GIA'
STATA SCELTA
SI
J = T(K)
non
K = K + 1
SI
K ≤ L − 1
NO
SI
D(T(L − 1), J ≥ C1
NO
C1 = D(T(L − 1), J)
L1 = J
J = J + 1
NO
J ≤ N
NO
RETURN

Figura 8.14

La subroutine ALGORITMO fa, anche lei, appello a due altri sottoprogrammi:

2500 ricerca della successiva città
2700 test se la soluzione trovata è migliore e se si immagazzinare questa soluzione.

```
20 REM  V$ RAPPRESENTA IL NOME DELLE CITTA'
25 REM  T = TABELLA DI LAVORO I CUI ELEMENTI CONTENGONO
30 REM      I NUMERI DELLE CITTA' DEL PERCORSO
35 REM  T1  CONTIENE I NUMERI DELLE CITTA'
40 REM      DELLA MIGLIORE SOLUZIONE TROVATA
45 REM  D = E' LA MATRICE DELLE DISTANZE O DEI COSTI
90 DIM V$(20),T(20),T1(20),D(20,20)
100 PRINT "PROGRAMMA DEL COMMESSO VIAGGIATORE "
110 PRINT
120 READ S$
125 IF S$ = "SYM" THEN  GOSUB 995
126 IF S$ <  > "SYM" THEN  GOSUB 800
130 GOSUB 1500
140 GOSUB 2000
150 GOSUB 3000
780 STOP
790 REM  LETTURA DELLA TABELLA DEI COSTI QUANDO LA MARICE NON E' SIMMETRICA
800 READ N
810 FOR I = 1 TO N
820  READ V$(I)
825  NEXT I
827 FOR I = 1 TO N
830  FOR J = 1 TO N
840   READ D(I,J)
850   NEXT J
860  NEXT I
870 RETURN
990 REM  LETTURA DELLA TABELLA DEI COSTI DI MATRICE SIMMETRICA
995 READ N
1000 FOR I = 1 TO N
1010  READ V$(I)
1020  NEXT I
1030 FOR I = 1 TO N
1040  D(I,I) = 0
1050  FOR J = I + 1 TO N
1060   READ D(I,J)
1070   D(J,I) = D(I,J)
1080   NEXT J
1090  NEXT I
1100 RETURN
1480 REM
1490 REM  SUBROUTINE PER STAMPARE LA MATRICE DEI COSTI
1500 PRINT "TABELLA DEI COSTI"
1510 PRINT
1520 FOR I = 1 TO N
1530  PRINT  TAB( 5 * I);V$(I);
1540  NEXT I
1550 PRINT
1555 PRINT
1560 FOR I = 1 TO N
1570  PRINT V$(I);
1580  FOR J = 1 TO N
1590   PRINT  TAB( 5 * J);D(I,J);
1600   NEXT J
1605  PRINT
1608  PRINT
1610  NEXT I
1620  RETURN
1970 REM
```

Figura 8.15

```
1970 REM
1980 REM  INIZIO DELL'ALGORITMO
1990 REM
2000 S = 1E38
2002 FOR I = 1 TO N
2005  C = 0
2010  T(I) = I
2020  FOR L = 2 TO N
2030   GOSUB 2500
2040   T(L) = L1
2050   C = C + C1
2060   NEXT L
2065  C = C + D(T(N),T(1))
2070  GOSUB 2700
2090 NEXT I
2110 RETURN
2470 REM
2480 REM  RICERCA DELLA CITTA' SUCCESSIVA
2490 REM
2500 C1 = 1E38
2510 FOR J = 1 TO N
2515  FOR K = 1 TO L - 1
2520   IF T(K) = J THEN 2560
2525   NEXT K
2530  IF D(T(L - 1),J) >  = C1 THEN 2560
2540   C1 = D(T(L - 1),J)
2550   L1 = J
2560  NEXT J
2570 RETURN
2670 REM
2680 REM  LA SOLUZIONE TROVATA E' LA MIGLIORE ?
2690 REM  SE SI, TRASFERIRE T IN T1 E C IN S
2700 IF S <  = C THEN 2750
2710 S = C
2720 FOR K = 1 TO N
2730  T1(K) = T(K)
2740  NEXT K
2750 RETURN
3000 PRINT
3010 PRINT  TAB( 9);"SOLUZIONE"
3015 PRINT
3020 FOR L = 1 TO N - 1
3030  PRINT V$(T1(L));"  A  ";V$(T1(L + 1)),D(T1(L),T1(L + 1))
3040  PRINT
3050  NEXT L
3055 PRINT V$(T1(N));"  A  ";V$(T1(1)),D(T1(N),T1(1))
3060 PRINT
3070 PRINT "COSTO TOTALE : "; TAB( 32);S
3080 RETURN
5000 END
```

Annotazioni:
- righe 2060–2070: aggiungere il costo di ritorno della città di partenza
- righe 2515–2520: la città seguente non deve essere già stata scelta
- righe 2530–2540: si considera la più vicina

Figura 8.16

PROGRAMMA DEL COMMESSO VIAGGIATORE

TABELLA DEI COSTI

	VR	BS	PV	TR	CR	BG	PC	MI
VR	0	45	67	13	40	68	89	81
BS	47	0	29	37	22	23	41	36
PV	68	30	0	73	21	24	12	37
TR	13	36	74	0	42	60	95	73
CR	40	24	22	43	0	36	33	49
BG	67	23	25	60	35	0	36	13
PC	89	40	13	98	35	36	0	47
MI	81	36	37	75	48	15	46	0

SOLUZIONE

```
TR  A   VR   13
VR  A   CR   40
CR  A   PV   22
PV  A   PC   12
PC  A   BG   36
BG  A   MI   13
MI  A   BS   36
BS  A   TR   37
COSTO TOTALE :   209
```

Dati corrispondenti:

```
155  DATA  "SYM",8
160  DATA  REM  LETTURA DELLA TABELLA DEI COSTI QUANDO LA MARICE NON E' SIMMETRI
170  DATA  45,67,13,40,68,89,81
180  DATA  29,37,22,23,41,36
190  DATA  73,21,24,12,37
200  DATA  42,60,95,73
210  DATA  36,33,49
220  DATA  36,13
230  DATA  47
```

Figura 8.17

```
PROGRAMMA DEL COMMESSO VIAGGIATORE

TABELLA DEI COSTI
```

	VR	BS	PV	TR	CR	BG	PC	MI
VR	0	45	67	13	40	68	89	81
BS	45	0	29	37	22	23	41	36
PV	67	29	0	73	21	24	12	37
TR	13	37	73	0	42	60	95	73
CR	40	22	21	42	0	36	33	49
BG	68	23	24	60	36	0	36	13
PC	89	41	12	95	33	36	0	47
MI	81	36	37	73	49	13	47	0

```
        SOLUZIONE

TR  A       VR      13
VR  A       CR      40
CR  A       PV      21
PV  A       PC      12
PC  A       BG      36
BG  A       MI      13
MI  A       BS      36
BS  A       TR      37

COSTO TOTALE : 208
```

Dati corrispondenti:

```
155  DATA  "SYM",8
160  DATA  VR,BS,PV,TR,CR,BG,PC,MI
170  DATA  45,67,13,40,68,89,81
180  DATA   29,37,22,23,41,36
190  DATA   73,21,24,12,37
200  DATA   42,60,95,73
210  DATA   36,33,49
220  DATA   36,13
230  DATA   47
```

Figura 8.18

Percorso corrispondente al risultato precedente:

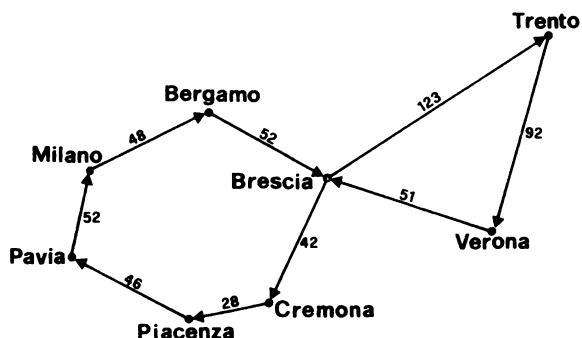



Figura 8.19

Questo risultato non corrisponde al circuito più corto che può essere trovato e il cui costo è di 206. Questo percorso di costo minimo è il seguente:

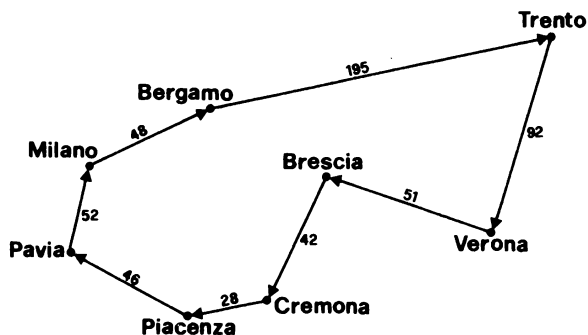



Figura 8.20

NOTE: 1. Se una città è "equidistante" da due altre città e se i costi sono minimi, l'algoritmo programmato non esegue test successivi con ciascuna delle città, ma sceglie sistematicamente la prima in ordine di indice.

Per obbligare l'algoritmo a effettuare il secondo test, occorre aumentare artificialmente di poco il costo del primo tragitto e domandare una seconda esecuzione.

2. Per aumentare la velocità dell'algoritmo è possibile modificarlo, imponendogli dei tratti che fanno parte del percorso di costo minimo.

8.4 CONCLUSIONI

Si sono visti tre semplici programmi di ricerca operativa. Il lettore ha potuto rendersi conto della lunghezza di questi programmi nonostante la semplicità apparente del problema. In generale ogni volta che è necessario "percorrere un grafico" è necessario ricorrere a manipolazioni piuttosto delicate di indici che generano difficoltà di programmazione.

Si consiglia al lettore interessato a questo problema di documentarsi sui seguenti argomenti:

— l'algoritmo di Kruskal,
— il problema dell'ottimizzazione del trasporto (algoritmo dello "steppingstone"),
— l'ottimizzazione del flusso in un grafico (algoritmo di Ford-Fulkerson),
— programmazione lineare (algoritmo del simflesso).

CAPITOLO 9

STATISTICA

9.0 INTRODUZIONE

Per l'elaborazione di diverse statistiche si fa spesso riferimento a microcalcolatori le cui qualità (velocità di calcolo, accesso ad un gran numero di dati) li rendono particolarmente adatti a questo tipo di applicazioni.

In questo capitolo si studieranno dei programmi di statistica semplici, ma molto utili. Per esempio per il calcolo del tasso di crescita studiato al Capitolo 6, si fa riferimento a un sottoprogramma di regressione lineare spiegato in questo capitolo.

Il numero di esercizi presentati è limitato al fine di non annoiare il lettore che non sia interessato a questo tipo di programmi. E' necessario sapere, tuttavia, che esiste un gran numero di programmi per questo campo.

9.1 MEDIA DI UNA SERIE DI MISURE

Calcolare la media aritmetica M di una serie di misure. In questo esercizio si suppone che tutti i dati siano incorporati nel programma. Un valore numerico uguale a —999 indica la fine dei dati.

Problema: Analizzare il problema, disegnare il flow-chart e scrivere quindi il programma.

Soluzione: Ogni misura è utilizzata una sola volta per il calcolo nel corso della somma. Non è dunque necessario utilizzare una tabella. Il numero di misure sarà registrato in una variabile N al fine di poter effettuare la divisione.

Alfine di non sommare il valore —999, è necessario effettuare un test A = —999

se A ≠ —999 effettuare la somma: M = M + A
N = N + 1
se A = —999 tutti i dati sono stati posizionati ed è necessario effettuare la divisione M = M/N.

Questo condurrà al flow-chart di Fig.9.1.

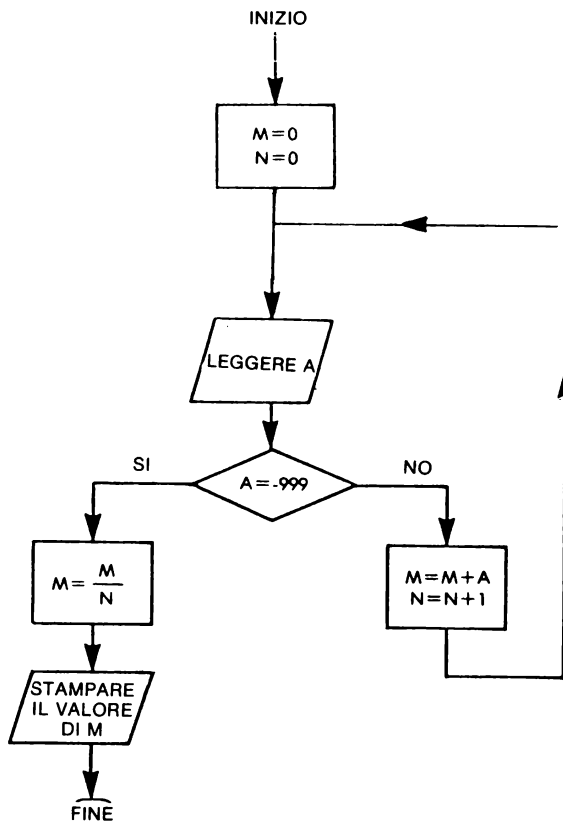



Figura 9.1

La programmazione di questo flow-chart è facile. La sola difficoltà risiede nel curare la presentazione dei risultati.

```
100 M = 0:N = 0
110  READ A
120  IF A =  - 999 THEN 170
130 N = N + 1
140 M = M + A
150  GOTO 110
170 M = M / N
180  PRINT "NUMERO DELLE MISURE= ";N
190  PRINT
200  PRINT "MEDIA "; TAB( 20);"= ";M
210  DATA  12,25,15,0,-999
220  END
```

Figura 9.2

9.2 MEDIA, VARIANZA, DEVIAZIONE STANDARD

Per stimare la media, la varianza e la deviazione standard di una serie di N misure si possono utilizzare le seguenti formule:

Media $M = \frac{1}{N} \sum_{i=1}^{N} A(I)$

Varianza $V = \frac{1}{N-1} \sum_{i=1}^{N} (A(I) - \overline{M})^2$

Deviazione standard $E = \sqrt{V}$

Come precedentemente supposto i dati sono incorporati nel programma e il valore numerico —999 indica la fine dei dati (ricerca della fine dei dati).

9.2.1 Primo problema: Si tratta di partire da una serie di misure supposte incorporate nel programma per stimarne la media, la varianza e la deviazione standard per mezzo delle formule precedenti. Per questo viene proposto di trattare questo problema in tre parti.

Parte *a*: Disegnare il flow-chart che permette di ottenere il valore numerico delle tre stime.

Parte *b*: Modificare la formula che fornisce il valore di V affinchè il flow-chart abbia un solo loop di calcolo.

Parte *c*: Scrivere un programma corrispondente al secondo flow-chart.

Soluzione:

Parte *a*: Si è condotti a costruire il flow-chart di Fig.9.3 che consta di due parti:

— una prima parte destinata a calcolare la media,
— una seconda parte che permette di ottenere la varianza e la deviazione standard.

Questo dà il flow-chart di Fig.9.3 che comporta due loop di calcolo e due letture dei dati.

Nel caso in cui i dati siano poco numerosi e incorporati nel programma, questa doppia lettura di dati non costituisce una particolare difficoltà. Per contro nella pratica, quando si tratta di archivi di dati di una certa consistenza, una doppia

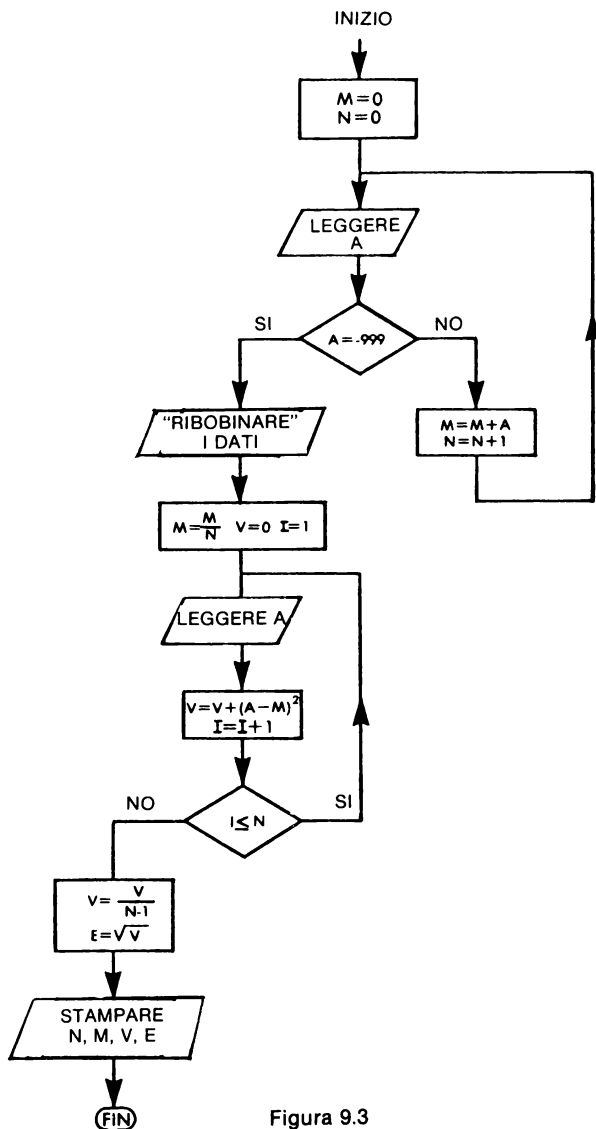



Figura 9.3

lettura raddoppia i tempi di calcolo se si è in "multiprogrammazione" e aumenta notevolmente il ritardo di restituzione se il microcalcolatore utilizzato lavora in "multiprogrammazione". Per questa ragione si prova a minimizzare il numero d'accessi all'archivio.

Parte *b*: Sviluppando la formula che fornisce il valore di V si ottiene:

$$V = \frac{1}{N^1}\left[\sum_{i=1}^{N} A(i)^2 - 2\overline{M}\sum_{i=1}^{N} A(i) + n\overline{M}^2\right]$$

con

$$\sum_{i=1}^{N} A(i) = N\overline{M}$$

da cui

$$V = \frac{1}{N-1}\left[\sum_{i=1}^{N} A(i)^2 - N\overline{M}^2\right]$$

Questa formula permette di calcolare M e V per mezzo di un solo loop di calcolo come si vede nel flow-chart che segue.

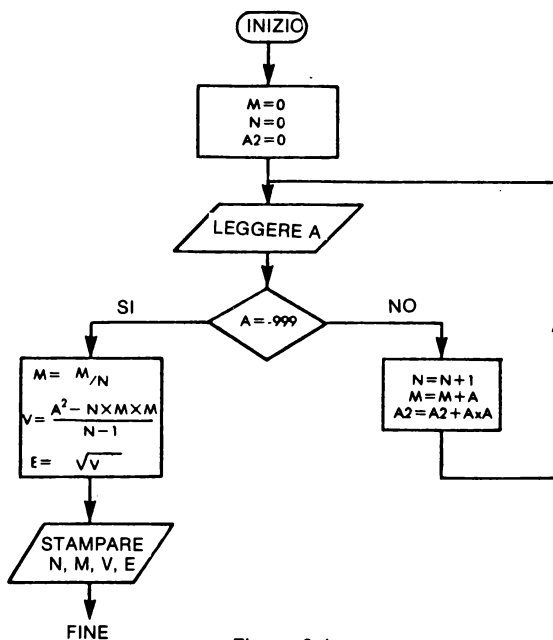



Figura 9.4

Parte *c*: Questo flow-chart è molto semplice e la programmazione è facile. La sola difficoltà consiste nel curare la presentazione dei risultati (Figg.9.5, 9.6).

```
100 M = 0
110 N = 0
120 A2 = 0
130  READ A
140  IF A =  - 999 THEN 190
150 N = N + 1
160 M = M + A
170 A2 = A2 + A * A
180  GOTO 130
190 M = M / N
200 V = (A2 - N * M * M) / (N - 1)
210 E =  SQR (V)
220  PRINT "NUMERO DELLE MISURE= ";N
230  PRINT "              MEDIA= ";M
240  PRINT "         VARIAZIONE= ";V
250  PRINT "DEVIAZIONE STANDARD= ";E
260  STOP
300  DATA   9,9.9,10,8.5,9;10.1
310  DATA  10,9.8,10.2
320  DATA  -999
330  END
```

Figura 9.5

Esempio di esecuzione:

```
NUMERO DELLE MISURE= 9
              MEDIA= 9.61111112
         VARIAZIONE= .373611014
DEVIAZIONE STANDARD= .611237281
```

Figura 9.6

9.2.2 Calcolo dei coefficienti d'asimetria e appiattimento

Modificare il programma precedente per calcolare il valore numerico delle stime seguenti:

Coefficiente di asimmetria $\frac{1}{NE^3} \sum_{I=1}^{N} (A(I) - \overline{M})^3$

Coefficiente di appiattimento $\frac{1}{NE^4} \sum_{I=1}^{N} (A(I) - \overline{M})^4$

Soluzione: Si noti che il momento d'ordine 2 si scrive:

$$M_2 = \frac{1}{N} \sum_{i=1}^{N} (A(i) - \overline{M})^2$$

Corrisponde a una stima per vie traverse della varianza.

Si pone

$$V_1 = \sum_{i=1}^{N} (A(I) - M)^2$$

Si avrà

$$V = \frac{1}{N-1} V_1$$

$$M_2 = \frac{1}{N} V_1$$

Si possono ora scomporre le due formule che danno S e P

$$S = \frac{1}{N\left(\frac{V_1}{N}\right)^{\frac{3}{2}}} [\Sigma\, A(i)^3 - 3\,\overline{M}\,\Sigma\, A(i)^2 + 3\,\overline{M}^2\,\Sigma\, A(i) - N\overline{M}^3]$$

$$= \frac{1}{N\left(\frac{V_1}{N}\right)^{\frac{3}{2}}} [\Sigma\, A(i)^3 - 3\,\overline{M}\,\Sigma\, A(i)^2 + 2 \times N \times \overline{M}^3]$$

$$P = \frac{1}{N\left(\frac{V_1}{N}\right)^{2}} [\Sigma\, A(i)^4 - 4\,\overline{M}\,\Sigma\, A(i)^3 + 6\,\overline{M}^2\,\Sigma\, A(i)^2 - 4\,\overline{M}^3\,\Sigma\, A(i) + NM^4]$$

$$= \frac{N}{V_1^{\,2}} [\Sigma\, A(i)^4 - 4\,\overline{M}\,\Sigma\, A(i)^3 + 6\,\overline{M}^2\,\Sigma\, A(i)^2 - 3\,N\overline{M}^4]$$

E' sufficiente ora inserire nel loop i calcoli di $\Sigma\, A(i)^3$ e di $\Sigma\, A(i)^4$ che saranno posti nelle variabili A3 e A4, per ottenere in seguito i coefficienti cercati.

$$S = \frac{1}{N\left(\frac{V_1}{N}\right)^{\frac{3}{2}}} [A3 - 3\,M\,A2 + 2\,NM^2]$$

$$P = \frac{N}{V_1^{\,2}} [A4 - 4\,M\,A3 + 6\,M^2\,A2 - 3\,NM^4]$$

La costruzione del programma di Fig.9.7 non presenta a questo punto più alcuna difficoltà.

```
100 A1 = 0
110 N = 0
120 A2 = 0
125 A3 = 0
127 A4 = 0
130  READ A
140  IF A =  - 999 THEN 190
150 N = N + 1
155 A1 = A1 + A
160 X = A * A
162 A2 = A2 + X
165 A3 = A3 + X * A
167 A4 = A4 + X * X
180  GOTO 130
190 M = A1 / N
200 V = (A2 - N * M * M) / (N - 1)
210 E =  SQR (V)
220  PRINT "NUMERO DELLE MISURE= ";N
230  PRINT "                MEDIA= ";M
240  PRINT "           VARIAZIONE= ";V
250  PRINT "DEVIAZIONE STANDARD= ";E
253 M2 = M * M
255 S = (A3 - 3 * M * A2 + 2 * M2 * A1) / (N * V * E)
260 P = (A4 - 4 * M * A3 + 6 * M2 * A2 - 3 * N * M2 * M2) / (N * V * V)
270  PRINT "    COEFFICIENTE DI SIMMETRIA= ";S
280  PRINT "COEFFICIENTE DI APPIATTIMENTO= ";P
285  STOP
300  DATA  2,2.5,3,3.5,4
310  DATA  -999
320  DATA  -999
330  END
```

Figura 9.7

Esempio di esecuzione:

```
NUMERO DELLE MISURE= 5
              MEDIA= 3
         VARIAZIONE= .625
DEVIAZIONE STANDARD= .790569415
    COEFFICIENTE DI SIMMETRIA= 0
COEFFICIENTE DI APPIATTIMENTO= 1.088
```

Figura 9.8

Altra serie di dati ad esecuzione corrispondente

```
NUMERO DELLE MISURE= 5
              MEDIA= 3
         VARIAZIONE= 2.5
DEVIAZIONE STANDARD= 1.58113883
    COEFFICIENTE DI SIMMETRIA= 0
COEFFICIENTE DI APPIATTIMENTO= 1.088
BREAK IN 285
```

Figura 9.8

NOTE:

1. Questi ultimi due sistemi di stima non sono valevoli con tutte le leggi. Devono quindi essere utilizzate con prudenza.
2. Coefficiente di simmetria vale 0 se il campione è simmetrico.
3. Coefficiente di appiattimento ingrandito nel valore assoluto con l'appiattimento della funzione di densità.

9.3 REGRESSIONE LINEARE

Cercare una retta che approssima nel migliore dei modi possibile l'insieme dei punti $(x;y)$ sperimentali. Per questo si utilizza generalmente il criterio dei "minimi quadrati" che consiste nel determinare i coefficienti a, b in modo che:

$$\sum_{i=1}^{N} (ax_i + b - y_i)^2$$

Per minimizzare questa quantità è necessario calcolare a e b come

$$a = \frac{n \, \Sigma \, x_i \, y_i - (\Sigma \, x_i)(\Sigma \, y_i)}{n \, \Sigma \, x_i - (\Sigma \, x_i)^2}$$

$$b = \frac{n \, \Sigma \, x_i - a \, \Sigma \, x_i}{n \, \Sigma \, x_i^2 - (\Sigma \, x_i)^2}$$

Per assicurarsi della validità "statistica" del calcolo, si può calcolare il coefficiente r dato da

$$r = \text{segno di } b \ \sqrt{1 - \frac{\Sigma\,(Y_i - \hat{Y}_i)^2}{\Sigma\,(\hat{Y}_i - \overline{Y})^2}}$$

Se r è vicino a 1 il calcolo è valido dal punto di vista statistico; in caso contrario, la regressione lineare si applica piuttosto male alla serie dei dati. Si può allora calcolare una varianza di a e b e determinare l'intervallo limite di a e b.

Esercizio: Scrivere un sottoprogramma che partendo dalle tabelle T(100) e Y(100) determini una retta di regressione e calcoli il coefficiente r.

Il calcolo dei coefficienti A , B costituisce un sottoprogramma che inizia in 1000, e il calcolo del coefficiente R dev'essere realizzato in un sottoprogramma che inizia in 600.

Soluzione: La parte calcolo di A e B deriva dalle formule precedenti con l'ausilio di un loop di calcolo, si ottiene:

$$\Sigma \, x_i, \Sigma \, y_i, \Sigma \, x_i^2, \Sigma \, x_i \, y_i,$$

poi partendo da queste quantità determinare i valori di A e B di cui al flow-chart di Fig.9.9

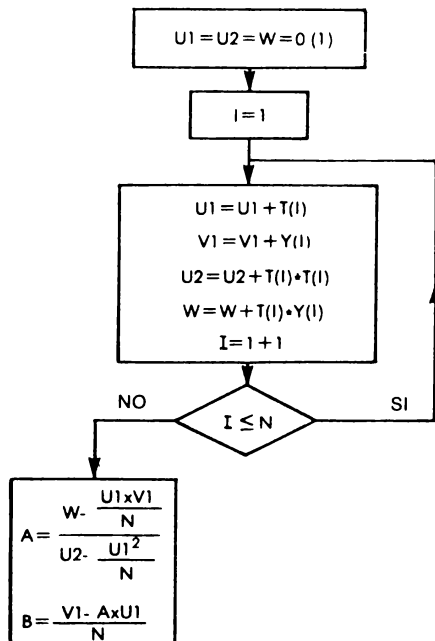



Figura 9.9

Per il calcolo di R si effettuerà un altro LOOP di calcolo

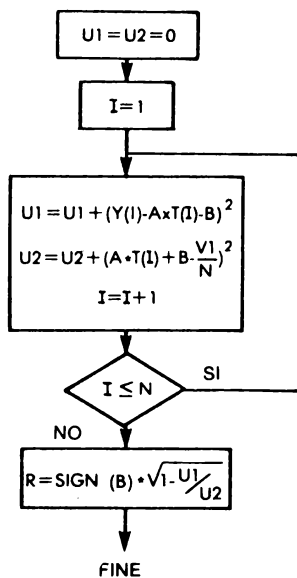



Figura 9.10

Partendo da questi due flow-chart, scrivere il programma con l'esempio di esecuzione di Fig.9.11 è cosa facile.

```
100   DIM T(100),Y(100)
110   READ N
120   FOR I = 1 TO N
130    READ T(I),Y(I)
140    NEXT I
150   GOSUB 1000
160   PRINT  TAB( 9);"INCLINAZIONE = ";A
170   PRINT "ORDINATA ALL'ORIGINE = ";B
180   PRINT
190   PRINT "T","MISURA Y"."CALCOLO Y"
200   PRINT
210   FOR I = 1 TO N
220    Y1 = A * T(I) + B
230    PRINT T(I),Y(I),Y1
240    NEXT I
245   STOP
250   DATA  5
260   DATA  0,1,1,1.5,2,2,4,3,6,4


1000 U1 = 0
1010 U2 = 0
1020 V1 = 0
1030 V2 = 0
1040 W = 0
1050 FOR I = 1 TO N
1060  U1 = U1 + T(I)
1070  V1 = V1 + Y(I)
1080  U2 = U2 + T(I) * T(I)
1090  V2 = V2 + Y(I) * Y(I)
1100  W = W + T(I) * Y(I)
1110  NEXT I
1120 A = (W - U1 * V1 / N) / (U2 - U1 * U1 / N)
1130 B = (V1 - A * U1) / N
1140 RETURN
1200 END


        INCLINAZIONE = .5
ORDINATA ALL'ORIGINE = 1

T               MISURA Y        CALCOLO Y

0               1               1
1               1.5             1.5
2               2               2
4               3               3
6               4               4
```

Figura 9.11

Regressione lineare senza calcolo del coefficiente *r*.

```
100 DIM T(100),Y(100)
110 READ N
120 FOR I = 1 TO N
130  READ T(I),Y(I)
140  NEXT I
150 GOSUB 1000
155 GOSUB 600
160 PRINT  TAB( 9);"INCLINAZIONE = ";A
170 PRINT "ORDINATA ALL'ORIGINE = ";B
175 PRINT  TAB( 7);"COEFFICIENTE R = ";R
180 PRINT
190 PRINT "T","MISURA Y","CALCOLO Y"
200 PRINT
210 FOR I = 1 TO N
220  Y1 = A * T(I) + B
230  PRINT T(I),Y(I),Y1
240  NEXT I
245 STOP
250 DATA  5
260 DATA  0,1,1,1.5,2,2,4,3,6,4
600 U1 = 0
605 U2 = 0
610 FOR I = 1 TO N
620  U1 = U1 + (Y(I) - A * T(I) - B) ^ 2
630  U2 = U2 + (A * T(I) + B - V1 / N) ^ 2
640  NEXT I
650 R =  SGN (B) *  SQR (1 - U1 / U2)
660 RETURN

1000 U1 = 0
1010 U2 = 0
1020 V1 = 0
1040 W = 0
1050 FOR I = 1 TO N
1060  U1 = U1 + T(I)
1070  V1 = V1 + Y(I)
1080  U2 = U2 + T(I) * T(I)
1100  W = W + T(I) * Y(I)
1110  NEXT I
1120 A = (W - U1 * V1 / N) / (U2 - U1 * U1 / N)
1130 B = (V1 - A * U1) / N
1140 RETURN
1200 END
```

Figura 9.12

Il programma della figura 9.12 corrisponde al calcolo dei coefficienti *a* e *b* e del coefficiente *r*.

Esempi di esecuzione con diversi set di dati.

```
         INCLINAZIONE = .5
ORDINATA ALL'ORIGINE = 1
       COEFFICIENTE R = 1

T              MISURA Y       CALCOLO Y

0              1              1
1              1.5            1.5
2              2              2
4              3              3
6              4              4
```

```
          INCLINAZIONE = .503125
ORDINATA ALL'ORIGINE = 1.00312
      COEFFICIENTE R = .998166

T               MISURA Y        CALCOLO Y

0               .95             1.00312
1               1.55            1.50625
2               2.05            2.00937
4               2.95            3.01562
4               3.05            3.01562

          INCLINAZIONE = .480603
ORDINATA ALL'ORIGINE = 1.25043
      COEFFICIENTE R = .932274

T               MISURA Y        CALCOLO Y

0               .95             1.25043
1               1.55            1.73103
2               2.95            2.21164
4               3.05            3.17284
6               4               4.13405
```

Questo esempio mostra la sensibilità del coefficiente di correlazione utilizzato.

9.4 CONCLUSIONE

Questi esercizi mostrano che la programmazione di calcoli elementari del tipo media, varianza, etc. non presenta particolari difficoltà. L'esercizio di calcolo di una regressione lineare è molto utile poichè viene utilizzato nel Capitolo 6 per il calcolo del tasso di crescita.

Invece quando è necessario addentrarsi in calcoli più sofisticati (prove statist che, regressione multipla, regressione polinomiale, etc.), i programmi si allungano diventano sensibili agli errori di arrotondamento.

Il lettore troverà in libri specializzati degli esercizi supplementari:

— istogrammi,
— regressione lineare multipla.

CAPITOLO 10

VARI

10.0 INTRODUZIONE

Questo capitolo contiene un insieme di esercizi che non fanno parte di nessun argomento trattato nei capitoli precedenti, ma che presentano un interesse sul piano informatico sia perchè fanno appello a delle astuzie di programmazione, sia perchè l'elaborazione del flow-chart non è evidente.

10.1 I SEGNI ZODIACALI

Partendo dal giorno e dal mese di nascita, determinare il "segno zodiacale" al quale si appartiene. Questo è facilmente realizzabile in quanto è sufficiente consultare la tabella 10.1 per ottenere la risposta.

SEGNO	PERIODO
CAPRICORNO	23 dic. al 19 gen.
AQUARIO	20 gen. al 19 feb.
PESCI	20 feb. al 20 mar.
ARIETE	21 mar. al 19 apr.
TORO	20 apr. al 20 mag.
GEMELLI	21 mag. al 20 giu.
CANCRO	21 giu. al 21 lug.
LEONE	22 lug. al 22 ago.
VERGINE	23 ago. al 23 set.
BILANCIA	23 set. al 22 ott.
SCORPIONE	23 ott. al 21 nov.
SAGITTARIO	22 nov. al 22 dic.

Figura 10.1

Esercizio 1:

Scrivere un programma che partendo dal giorno e mese di nascita determini il segno zodiacale corrispondente.

Si supporrà di disporre di un BASIC che accetti le tabelle costituite da stringhe di caratteri.

Esercizio 2:

Come il precedente ma disponendo questa volta di un BASIC che non accetta le tabelle costituite da stringhe di caratteri.

Soluzione esercizio 1:

Si devono confrontare il giorno del mese J con un limite L che dipende dal mese (L da 20 a 23).

- — Se J < L l'indice da utilizzare è I = M.
- — Se J ⩾ L l'indice da utilizzare è I = M + 1 salvo che questo dia I = 13 in tal caso è necessario attribuire a I il valore 1 (caso di una persona nata tra il 23 e il 31 dicembre).

Perchè L assuma il valore corretto si dovrà prima attribuirgli il valore 20 poi utilizzare un'istruzione ON...GOTO per effettuare il salto a una serie di istruzioni L = L + 1.

In questo modo si eviteranno parecchie prove e istruzioni GOTO.
Questo metodo può essere rappresentato dal flow-chart di Fig.10.2 che si presenta più difficile del programma di Fig.10.3 corrispondente.

Soluzione esercizio 2:

Il programma precedente determina un indice J che indica il numero del segno, ma questo indice non può essere utilizzato in questa forma.
I segni zodiacali più lunghi da scrivere sono CAPRICORNO e SAGITTARIO con 10 lettere. Si potrà quindi utilizzare una variabile, stringa di caratteri, A$ con lunghezza 120 che potrà contenere i dodici segni di 10 caratteri (quelli di lunghezza inferiore saranno completati con l'inserimento di spazi).

Partendo dall'indice I ottenuto precedentemente si scriverà:

MID$ (A$, 10 X I — 9,10)

Occorre ora poter inizializzare correttamente A$ per questo è possibile utilizzare diversi metodi, per esempio:

```
115 DIM A$(120)
120 A$="CAPRICORNOACQUARIO^^^PESCI ^^^ ARIETE^^^^TORO ^^^ "
125 A$=A$+"GEMELLI^^^^CANCRO^^^^LEONE^^^^^VERGINE^^^^BILANCIA^^^"
130 A$=A$+"SCORPIONE^^SAGITTARIO"
```

Ogni segno completandolo con un numero di caratteri "spazio", inizia nella giusta posizione (1,11,21,31 etc.) e potrà essere facilmente selezionato.

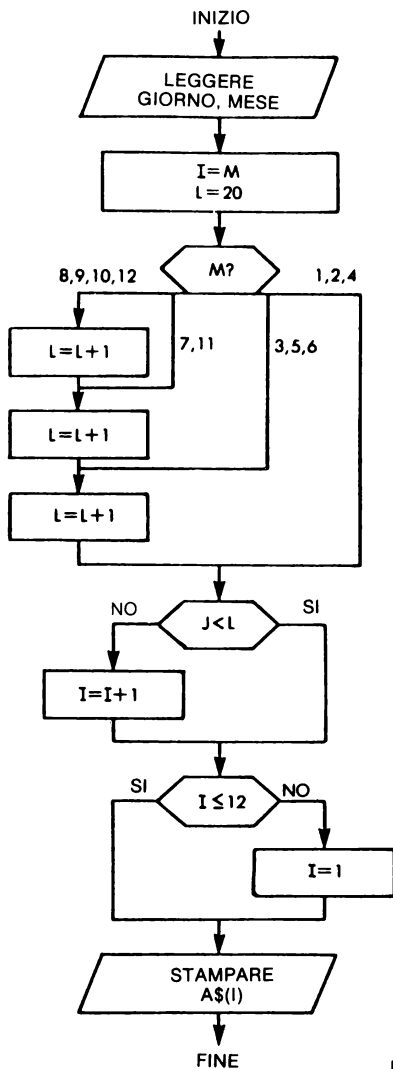



Figura 10.2

```
115 DIM A$(12)
120 FOR I = 1 TO 12
125  READ A$(I)
130  NEXT I
140 PRINT "GIORNO,MESE DI NASCITA ";
145 INPUT J,M
180 I = M
190 L = 20
200 ON M GOTO 600,600,500,600,500,500,400,300,300,300,400,300
300 L = L + 1
400 L = L + 1
500 L = L + 1
600 IF J < L THEN 610
605 I = I + 1
610 IF I <  = 12 THEN 620
615 I = 1
620 PRINT "VOI SIETE ";A$(I);
630 PRINT
650 GOTO 140
900 DATA  "CAPRICORNO","AQUARIO","PESCI","ARIETE"
910 DATA  "TORO","GEMELLI","CANCRO","LEONE","VERGINE"
920 DATA  "BILANCIA","SCORPIONE","SAGITTARIO"
990 END
```

Figura 10.3

```
GIORNO,MESE DI NASCITA ?12,10
VOI SIETE BILANCIA

GIORNO,MESE DI NASCITA ?19,1
VOI SIETE CAPRICORNO

GIORNO,MESE DI NASCITA ?20,1
VOI SIETE AQUARIO

GIORNO,MESE DI NASCITA ?21,5
VOI SIETE GEMELLI

GIORNO,MESE DI NASCITA ?23,9
VOI SIETE BILANCIA

GIORNO,MESE DI NASCITA ?
?REENTER
?
```

Figura 10.4

Attenzione questo programma non si ferma da solo!

10.2 REGINE SU UNA SCACCHIERA

Un problema classico per il gioco degli scacchi è il seguente: trovare tutte le soluzioni che permettono di piazzare 8 regine su una scacchiera senza che vi siano due regine sulla stessa linea orizzontale, verticale e diagonale.

Problema:

Generalizzando una scacchiera N X N, impostare tutte le soluzioni che permettono di posizionare N regine senza che ci siano due regine "in presa" tra loro. Si farà variare N da 2 a 8.

Si elimineranno le soluzioni che si deducono per simmetria.

Metodo proposto:

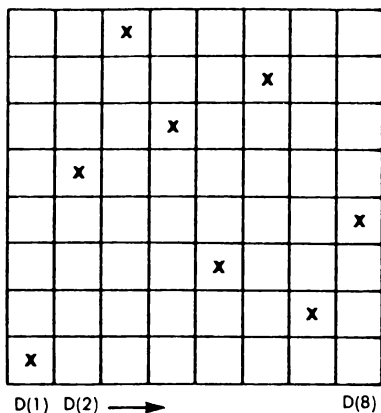



Figura 10.5

Una tabella D conterrà le regine posizionate nella soluzione in corso di elaborazione.

Per esempio la soluzione proposta in Fig.10.4 corrisponde a:

D(1) = 1	D(5) = 3
D(2) = 5	D(6) = 7
D(3) = 8	D(7) = 2
D(4) = 6	D(8) = 4

Con il modo di rappresentazione dei dati scelti, affinchè due regine non siano "in presa" occorre:

— che non siano sulla stessa colonna, e qui è realizzato poichè c'è solo una regina per colonna,

— che non siano sulla stessa linea, ciò significa che è necessario porre:

$$D(I) \neq D(J)$$

— che non siano sulla stessa diagonale. Per questo è necessario porre:

D(J) — D(I) ≠ J—I diagonale a 45 gradi
D(J) — D(I) ≠ I—J diagonale a —45 gradi

Questi due test possono essere riuniti nel seguente test:

$$\text{ABS}(D(I) - D(J)) \neq I - J$$

Criteri di ricerca delle soluzioni:

Si parte da D(1)=1.

Si cerca la prima posizione ammissibile per D(2), cioè una posizione tale per cui D(2) non sia "in presa" con D(1).

Si cerca allora una posizione ammissibile per D(3) e si continua fino a trovare una posizione per D(N).

Se non si trova la posizione ammissibile per D(I), si proverà a spostare D(I—1) soddisfacendo alle specifiche: D(I—1) non dovrà essere "in presa" con le regine precedenti (D(1),D(2)...D(I—2)) poi si cercherà una nuova posizione ammissibile per D(I).

Più precisamente l'algoritmo proposto è il seguente:

Per I che varia da 1 a N

1. Porre D(I) = 1
2. Verificare che la nuova regina D(I) non sia "in presa" con una delle regine già posizionate.
   Se è "in presa" andare in 3.
   Altrimenti, se I = N si ha una soluzione che si può stampare procedendo oltre (andare in 3).
   Se I < N porre I = I + 1 e andare in 1.
3. Cercare una nuova posizione per D(I):
   Porre D(I) = D(I) + 1
   Se D(I) ⩽ N andare in 2
   Se D(I) > N non ci sono posizioni convenienti: porre I=I—1, andare in 3.

Per eliminare le soluzioni che si deducono per simmetria da una soluzione già trovata:

a) Si farà variare D(1) da 1 a N/2 soltanto (si eliminano così tutte le soluzioni simmetriche rispetto all'asse orizzontale).

b) Non si stamperà alcuna soluzione D(1) > D(N) poichè questa soluzione corrisponde a una soluzione simmetrica rispetto all'asse verticale già stampata.

Il sottoprogramma di Fig. 10.6 rappresenta l'algoritmo descritto precedentemente.

Il flow-chart di Fig. 10.7 rappresenta il programma principale.

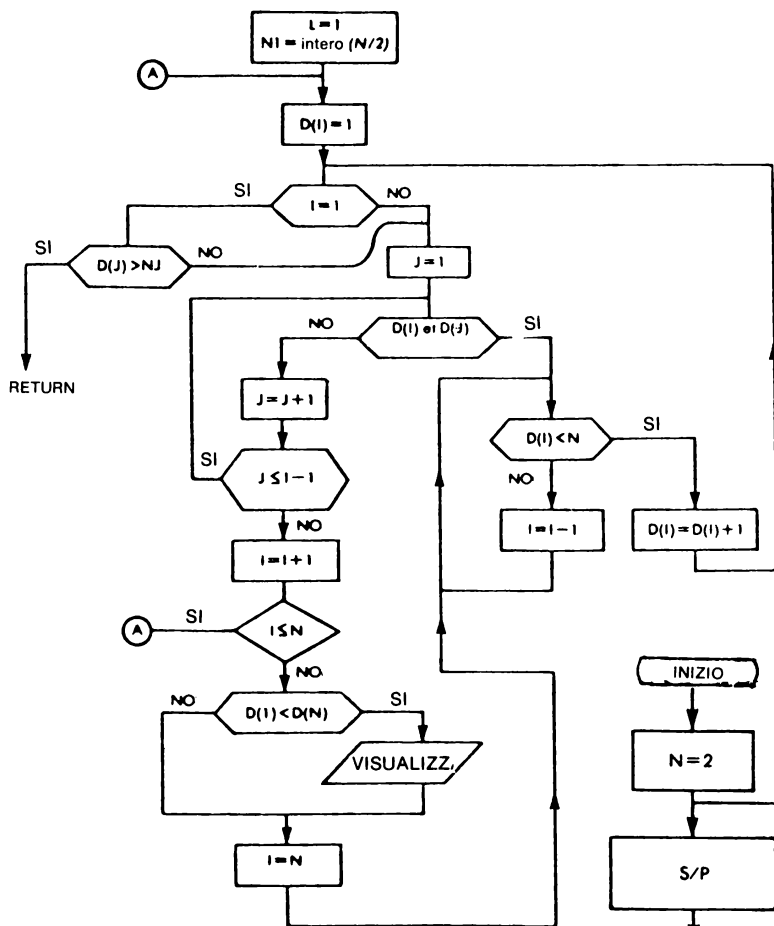



Figura 10.6

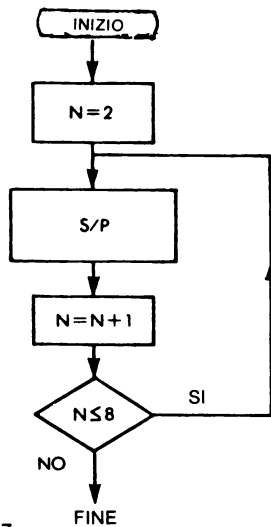



Figura 10.7

```
100 REM  RICERCA DEL NUMERO DELLE SOLUZIONI CHE PERMETTANO
110 REM  DI PORRE N REGINE SU UNA SCACCHIERA SENZA CHE  VE
120 REM  NE SIANO DUE "IN PRESA"
130 DIM D(10)
140 N9 = 8
150 FOR N = 2 TO N9
160  S = 0
170  PRINT
180  PRINT "N = ";N
190  PRINT
200  GOSUB 500
210  NEXT N
220 STOP
500 I = 1
510 N1 =  INT (N / 2)
520 D(I) = 1
530 IF I <  > 1 THEN 560
540 IF D(1) <  = N1 THEN 600
550 GOTO 690
560 FOR J = 1 TO I - 1
570  IF D(I) = D(J) THEN 640
580  IF  ABS (D(I) - D(J)) = I - J THEN 640
590  NEXT J
600 I = I + 1
610 IF I <  = N THEN 510
620 IF D(N) > D(1) THEN  GOSUB 700
630 I = N
640 IF D(I) < N THEN 670
650 I = I - 1
660 GOTO 640
670 D(I) = D(I) + 1
680 GOTO 530
690 RETURN
700 FOR L = 1 TO N
710  PRINT D(L); TAB( 4 * L);
720  NEXT L
730 PRINT
740 RETURN
750 END
]
```

Figura 10.8

Per i sistemi che dispongono di un BASIC che accetta le variabili intere (l'MBASIC della Microsoft ad esempio), la velocità di esecuzione sarà aumentata dichiarando tutte le variabili intere e sostituendo l'istruzione della linea 510 con N1=N/2.

10.3 CONCLUSIONI

Tutti questi esercizi hanno mostrato che la programmazione propriamente detta, non presenta in generale difficoltà, dopo aver acquisito la tecnica di effettuare

Esempio di esecuzione:

```
N=  2

N=  3

N=  4

 2   4   1   3

N=  5

 1   3   5   2   4
 1   4   2   5   3
 2   4   1   3   5
 2   5   3   1   4

N=  6

 2   4   6   1   3   5
 3   6   2   5   1   4

N=  7

 1   3   5   7   2   4   6
 1   4   7   3   6   2   5
 1   5   2   6   3   7   4
 1   6   4   2   7   5   3
 2   4   1   7   5   3   6
 2   4   6   1   3   5   7
 2   5   1   4   7   3   6
 2   5   3   1   7   4   6
 2   5   7   4   1   3   6
 2   6   3   7   4   1   5
 2   7   5   3   1   6   4
 3   1   6   2   5   7   4
 3   1   6   4   2   7   5
 3   6   2   5   1   4   7
 3   7   2   4   6   1   5
 3   7   4   1   5   2   6

N=  8

 1   5   8   6   3   7   2   4
 1   6   8   3   7   4   2   5
 1   7   4   6   8   2   5   3
 1   7   5   8   2   4   6   3
 2   4   6   8   3   1   7   5
 2   5   7   1   3   8   6   4
 2   5   7   4   1   8   6   3
 2   6   1   7   4   8   3   5
 2   6   8   3   1   4   7   5
 2   7   3   6   8   5   1   4
 2   7   5   8   1   4   6   3
 2   8   6   1   3   5   7   4
 3   1   7   5   8   2   4   6
 3   5   2   8   1   7   4   6
 3   5   7   1   4   2   8   6
 3   5   8   4   1   7   2   6
 3   6   2   5   8   1   7   4
 3   6   2   7   1   4   8   5
 3   6   2   7   5   1   8   4
 3   6   8   1   5   7   2   4
 3   6   8   2   4   1   7   5
 3   7   2   8   5   1   4   6
 3   7   2   8   6   4   1   5
 3   8   4   7   1   6   2   5
 4   1   5   8   2   7   3   6
 4   2   5   8   6   1   3   7
 4   2   7   3   6   8   1   5
 4   2   8   5   7   1   3   6
 4   2   8   6   1   3   5   7
 4   6   1   5   2   8   3   7
 4   6   8   2   7   1   3   5
 4   7   3   8   2   5   1   6
 4   7   5   2   6   1   3   8
 4   8   1   3   6   2   7   5
 4   8   5   3   1   7   2   6
```

l'analisi del problema e di ricavarne il relativo flow-chart. Questo è il caso dell'ultimo esercizio inerente alle regine sulla scacchiera.

Al termine di questo libro vogliamo dare un ultimo consiglio: prima di lanciarvi nella scrittura di un programma:

- **Cercate se tale programma esiste, in caso affermativo, se è possibile, procuratevelo.**
- **Prima di cominciare riflettere bene: il tempo passato nel fare l'analisi è presto riguadagnato durante la scrittura del programma e della sua messa a punto.**

APPENDICE I

L'ALFABETO IN BASIC

L'alfabeto in BASIC comprende:

— le lettere maiuscole da	A a Z
— i numeri da	0 a 9
— i simboli di operazione	
addizione e sottrazione	+ e —
moltiplicazione e divisione	× e /
elevamento a potenza(1)	↑
— le parentesi	(e)
— i caratteri di confronto	
uguale	=
minore	<
maggiore	>
— i caratteri di punteggiatura	
virgola e punto e virgola	, e ;
punto e due punti	. e :
punto esclamativo e interrogativo	! e ?
— i caratteri speciali	
"spazi"	
apostrofo	'
virgolette	"
dollaro	$

● Partendo da questo alfabeto si costruiscono i programmi in BASIC.

● Alcuni sistemi ammettono delle differenze o dei caratteri supplementari.

---

(1) Su alcuni sistemi questa freccia verticale è sostituita da un accento circonflesso.

LE COSTANTI E LE VARIABILI NUMERICHE

Costanti numeriche: Si può rappresentare un numero per mezzo di:

— un numero intero con o senza segno:

Esempio: 11, —162

— un numero decimale senza esponente:

Esempio: 3.1415927, —3. , 0.12 , .12

La parte intera è separata dalla parte decimale per mezzo di un punto . la virgola non viene utilizzata a questo scopo.

— un numero con esponente:

Esempio: 1E + 5 — 1.6 E — 19
esponente

Il primo segno è il segno del numero, il secondo è quello dell'esponente:

— 1.6 E —19 rappresenta —1,6 10^{-19}

Poichè si è costretti a scrivere i numeri su una linea, si separa l'esponente dal resto con la lettera E.

Attenzione: Il calcolatore distinque il numero 0 dalla lettera O maiuscola. L'utente che utilizza la tastiera deve fare la stessa distinzione.

LE VARIABILI NUMERICHE

Si distinguono due categorie di variabili numeriche:

— le variabili semplici,
— le variabili con indice dette anche "MATRICI".

Le variabili numeriche semplici sono caratterizzate dal loro "nome" (chiamato anche identificatore) che è rappresentato:

— sia da una lettera da A a Z,
— sia da una lettera seguita da un numero.

Esempio: A, B, R1, R9 sono delle variabili.

In un programma si può dunque avere un totale di 26 variabili a una lettera e 26 × 10 variabili costituite da una lettera seguita da un numero per complessive 286 variabili distinte.

NOTA: La maggior parte dei BASIC per microcalcolatori (soprattutto quando l'interprete è stato sviluppato dalla Microsoft) accetta dei nomi composti sia da una lettera, che da una lettera seguita da un carattere "alfanumerico" (lettera o numero).

Le variabili indice sono caratterizzate dal loro nome e da uno o due indici.

Esempio: A(R,I), B(I), C(I + 10 X K)

Gli indici possono essere delle costanti, delle variabili o delle espressioni aritmetiche.

Prima di utilizzare una variabile indice, occorre dichiararla per mezzo di un'istruzione DIM all'inizio del programma.

Esempio: DIM A(10, 20).

LE ESPRESSIONI ARITMETICHE

Le espressioni aritmetiche sono costruite a partire da:

— variabili e costanti,
— operatori +, —, *, /, ↑,
— funzioni numeriche standard
— funzioni dell'utilizzatore (numeriche),
— parentesi.

Le parentesi svolgono due funzioni:

— Inquadrare l'argomento o gli argomenti di una funzione.
— Impostare l'ordine col quale è necessario effettuare i calcoli per "valutare" il valore dell'espressione.

Esempio: A + B·C significa $a + (b \times c)$
(A + B)·C significa $(a + b) \times c$
A + B·SIN(C + 3) qui le parentesi "inquadrano" l'argomento che è C + 3.

Normalmente le operazioni sono effettuate tenendo conto della "priorità degli operatori" e a priorità uguale nell'ordine da sinistra a destra. L'ordine di priorità decrescente è il seguente:

parentesi,
funzione,
operatore elevamento a potenza,
moltiplicazione e divisione,
addizione e sottrazione

Esempio: l'espressione

è eseguita nell'ordine indicato e corrisponde a:

$$A \times (B + 3.2 \times \text{SIN}\,(y + 3\,z)) + \frac{x^4}{C}.$$

ISTRUZIONI DI ASSEGNAZIONE ARITMETICA

La forma di questa istruzione è:

variabile = espressione aritmetica
↑ variabile semplice o elemento di tabella

Il significato è: calcolare il valore numerico dell'espressione situata a destra del segno uguale, e racchiudere il risultato nella variabile situata a sinistra del segno uguale.

Esempio: V = 4 * 3.14159 × R ↑ 3/3
X = A

ISTRUZIONI DI SALTO

Salto "incondizionato"

La forma più semplice dell'istruzione GOTO è:

GO TO E

oppure

GOTO E

con E numero di linea.

Questa istruzione provoca il salto alla linea *n*.

Salto "calcolato"

Questa istruzione ha in generale la seguente forma:

ON espressione numerica GO TO E_1 ,E_2 ,E_3 ,E_p
con E_1 ,E_2 ,...,E_p numeri di linea.

Nell'esecuzione questa istruzione provoca la valutazione dell'espressione, l'eventuale conversione per arrotondamento in un intero e, quindi il salto

— alla linea E_1 se il valore ottenuto è 1
— alla linea E_2 se il valore ottenuto è 2
— alla linea E_p se il valore ottenuto è *p*.

Esempio: ON I GO TO 100, 200, 600, 200

se I vale 1 salto in 100
se I vale 2 o 4 salto in 200
se I vale 3 salto in 600

Se il valore dell'espressione dopo l'eventuale arrotondamento non cade in un'intervallo (1, *p*), il risultato è incerto:

— l'esecuzione prosegue in sequenza, oppure
— la segnalazione di un codice errore.

Salto condizionato

I test sono effettuati per mezzo dell'istruzione IF che può assumere differenti forme:

Prima forma: Si tratta della forma più semplificata:

IF { espressione di comparazione } THEN { numero di linea }

Se la risposta al confronto è si, vi è un salto alla linea indicata, altrimenti l'esecuzione prosegue in sequenza.

Esempio: IF A < B THEN 600

Il confronto si può effettuare con i seguenti simboli:

minore	<
maggiore	>
minore o uguale	<=
maggiore o uguale	>=
diverso da	<>

Questi operatori possono essere utilizzati con delle variabili numeriche e con stringhe di caratteri.

Seconda forma: Si tratta di una forma più efficiente:

IF { espressione di comparazione } THEN Istruzione eseguibile
↑
questa istruzione non può essere un'istruzione FOR

— Se la risposta al test è SI allora l'istruzione è eseguita e se questa istruzione non è un salto, l'esecuzione prosegue in sequenza.

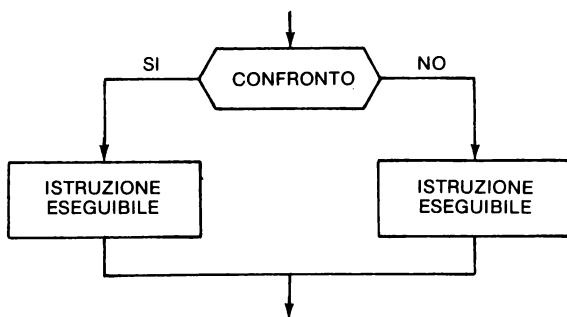
— Se la risposta al test è NO l'istruzione non è eseguita, vi è il passaggio alla linea seguente.

Esempio: IF A < B THEN X=B
IF A < B THEN GO TO 600

Terza forma: Questa forma molto evoluta è di più facile utilizzazione.

IF { espressione di comparazione } THEN { istruzione eseguibile } ELSE { istruzione eseguibile }

Esempi:

IF A < B THEN C = A : D = B ELSE C = B : D = A
IF A$ = B$ THEN I = I + 1 ELSE GO TO 100

Questa istruzione corrisponde al flow-chart sopra riportato.

Per alcuni sistemi ogni alternativa di un'istruzione IF può contenere una sola istruzione.

LOOP DI CALCOLO

Si realizza un loop di calcolo per mezzo dell'istruzione FOR ... NEXT.

FOR *var* = E1 TO E2 STEP E3
o
o
o
o
NEXT *var*

nella quale *var* è il nome di una variabile numerica:

E1, E2 e E3 rappresentano delle espressioni aritmetiche
E1 dà il valore di partenza
E2 dà il valore finale
E3 dà il passo
Se E1 < E2 deve essere E3 > 0
E1 > E2 deve essere E3 < 0

i valori numerici di E1, E2 e E3 sono calcolati prima di "incominciare" il loop di calcolo.
Se il passo è 1 si può scrivere

FOR *var* = E1 TO E2

Esempio:
```
100 FOR I = 1 TO 10
110 FOR J = 1 TO 10
120 A(I,J) = 0
130 NEXT J
140 NEXT I
```

STRINGHE DI CARATTERI

Costanti di caratteri

Sequenze di caratteri fra virgolette.

Esempio: "ABCD"
"BELLA DONNA"

Il carattere "spazio" è significativo in una stringa di caratteri. La lunghezza massima di una stringa di caratteri dipende dal sistema utilizzato.

Variabili di caratteri

Il nome di una variabile è definito da una lettera seguita dal carattere dollaro

Esempio: A$, B$

dunque un massimo di 26 variabili distinte in un programma.

Operazioni possibili sulle stringhe di caratteri

Confronti: Si dice che A$ minore di B$ se nell'ordine alfabetico la variabile A$ precede B$.

Esempio: A$ = "GIANNI"
B$ = "ROSSI"

qui B$ minore di A$
Concatenazione: consiste nell'affiancare due stringhe di caratteri.

Esempio: A$ = "PIETRO"
B$ = " VERDI"
N$ = A$ + B$ il carattere + è utilizzato in BASIC
come simbolo di concatenazione

N$ assume il valore "PIETRO∧VERDI".

Funzioni speciali per stringhe di caratteri: L'elenco delle funzioni non è limitativo e può variare da un sistema all'altro. La seguente lista rappresenta numerose funzioni disponibili sui microcalcolatori.

ASC (X$)	fornisce il valore numerico del primo carattere di X$ secondo la rappresentazione del codice ASCII (codice ISO) per esempio ASC ("A") dà 65.
CHR$(I)	fornisce una stringa di caratteri che rappresenta in ASCII il valore numerico di I.
IN$TR(I,A$,B$)	cerca la prima applicazione di B$ in A$ partendo dalla I-esema posizione. Fornisce la posizione della sottostringa B$ trovata.

LEN(A$)	fornisce la lunghezza della stringa A$
CHR$(I)	fornisce una stringa di caratteri che rappresenta, in codice ASCII, il valore di I.
HEX$(X)	fornisce una stringa di caratteri che rappresenta in esadecimale il valore decimale dell'argomento X.
LEFT$(A$,I)	fornisce una stringa di caratteri contenente gli I caratteri più a sinistra di A$
MID$(A$,I,J)	fornisce una stringa estratta da A$ partendo dal I-esimo carattere e di lunghezza J.
OCT$(I)	fornisce una stringa che rappresenta in codice ottale il valore decimale dell'argomento I.
RIGHT$(A$,I)	fornisce una stringa contenente gli I caratteri situati più a destra nella stringa A$
SPACE$(I)	fornisce una stringa contenente gli I caratteri "spazio".
STRING$(I,J)	fornisce una stringa di lunghezza I e rappresenta il valore numerico di J.
VAL(A$)	fornisce un numero che contiene il valore numerico della stringa ASCII A$. Questa funzione presuppone che i caratteri contenuti in A$ rappresentino un numero.

INPUT/OUTPUT

Lettura di dati "conversazionali":

```
INPUT    lista di variabili
          └─variabili semplici o elementi di tabelle
```

Lettura dei dati inclusi nel programma:

READ	lista delle variabili
DATA	valori numerici separati da una virgola (o da uno spazio in certi sistemi)
RESTORE	per "riavvolgere".

Visualizzazione dei risultati:

Prima forma:
PRINT lista
delle espressioni
delle costanti

Esempio: 100 PRINT "X = " ; X," Y = " ; Y

I separatori da utilizzare sono la virgola e il punto e virgola. La virgola non limita la lunghezza della zona utilizzata per la stampa. Il punto e virgola limita questa zona alla lunghezza necessaria per la spampa.

Un separatore alla fine dell'istruzione PRINT impedisce il passaggio alla linea seguente.

Nella lista, si può inoltre utilizzare, la funzione TAB che permette di inserire delle tabulazioni.

BIBBLIOGRAFIA

Flow-chart — norma AFNOR Z67-00 Aprile 1966

BASIC norma — proposta per una norma "minimal BASIC" ANSI Marzo 1979

libri — Il linguaggio BASIC e le sue estensioni, di J.P Lamoitier, editore Eyrolles
BASIC PROGRAMMING, di John G.Kemeny e Thomas E.Kurtz, editore John Wiley

giochi — BASIC COMPUTER GAMES, Microcomputer Edition, distribuito in Europa dalla Sybex.

Manuali di: — MICROSOFT

Manuali di costruttori di microcalcolatori: — PET, TRS-80, APPLE, CROMEMCO

Altri libri editi dal Gruppo Editoriale Jackson:

Esperimenti su circuiti logici e di memoria TTL-I, di David G. Larsen e Peter R. Rony - Codice 001A - L. 22.000

Esperimenti su circuiti logici e di memoria TTL-II, di David G. Larsen e Peter R. Rony - Codice 002A - L. 22.000

Interfacciamento e trasmissione dati UART e Loop a 20 mA, di David G. Larsen e Peter R. Rony - Codice 021A - L. 4.500

IL BUGBOOK III - Interfacciamento e programmazione del microcomputer 8080, di Peter Rony David Larsen Jonathan A. Titus - Codice 003A - L. 19.000

Interfacciamento di microcomputer - Esperimenti utilizzanti Chip 8255 PP I, di Paul Goldsbrough - Codice 004A - L. 10.500

Esperimenti con TTL e 8080A, Vol. 1, di Peter Rony - Codice 005A - L. 19.000

Esperimenti con TTL e 8080A, Vol. 2, di Peter Rony - Codice 006A - L. 19.000

IL BUGBOOK VII - L'interfacciamento tra microcomputer e convertitori analogici, di J. Titus / P. Rony / C. Titus / D. Larsen - Codice 007A - L. 15.000

Corso di elettronica fondamentale con esperimenti, di Siemens - Codice 201A - L. 15.000

Comprendere l'elettronica a stato solido, di Texas Instruments - Codice 202A - L. 14.000

Introduzione pratica all'impiego dei circuiti integrati digitali, di F. Hure - Codice 203A - L. 7.000

Elettronica integrata digitale, di H. Taub e D. Schilling - Codice 204A - L. 34.500

SC/MP, di A. Cavalcoli e V. Scibilia - Codice 301D - L. 9.500

Lessico dei microprocessori, di R. Zaks e A. Lesea - Codice 302P - L. 3.500

Introduzione al personal e business computing, di R. Zaks - Codice 303D - L. 14.000

Introduzione ai microcomputer - Il libro dei principianti, Vol. 0, di A. Osborne - Codice 304A - L. 14.000

Introduzione ai microcomputer - Il libro dei concetti fondamentali, Vol. 1, di A. Osborne - Codice 305A - L. 16.000

Practical microprocessors, di M. Slater e B. Bronson - Codice 308B - L. 35.000

Principi e tecniche di elaborazione dati, di NCR Corporation - Codice 309A - L. 15.000

NANOBOOK Z80, Vol. 1 - Tecniche di programmazione, di E. Nichols J. Nichols P. Rony - Codice 310P - L. 15.000

NANOBOOK Z80 - Vol. 3 - Tecniche di interfacciamento, di E. Nichols J. Nichols P. Rony - Codice 312P - L. 18.000

DBUG, di J. Titus e C. Titus - Codice 313P - L. 6.000

Tecniche di interfacciamento dei microprocessori, di R. Zaks e A. Lesea - Codice 314P - L. 22.000

Microelettronica: la nuova rivoluzione industriale, di A. Osborne - Codice 315P - L. 9.000

Elementi di trasmissione dati, di National Cash Register Co. - Codice 316D - L. 9.000

I microprocessori, di R. Zaks - Codice 320P - L. 22.000

La programmazione dello Z8000, di R. Matheasian - Codice 321D - L. 22.000

TEA, un editor assembler residente per l'8080/8085, di C. Titus - Codice 322P - L. 12.000

8080A/8085 - Programmazione in linguaggio assembly, di L. Leventhal - Codice 323P - L. 24.000

Programmazione dello Z80 e progettazione logica, di A. Osborne / J. Kane , R. Rector S. Jacobson - Codice 324P - L. 19.000

Programmazione dell'8080 e progettazione logica, di A. Osborne - Codice 325P - L. 16.500

Z80 programmazione in linguaggio assembly, di L. Leventhal - Codice 326P - L. 29.500

Usare il microprocessore, di G. Giaccaglini - Codice 327A - L. 15.000

Pascal - Manuale e standard del linguaggio, di K. Jensen e K. Wirth - Codice 500P - L. 10.000

Impariamo il Pascal, di F. Woldner - Codice 501A - L. 10.000

Introduzione al Basic, di P. Le Beux - Codice 502A - L. 18.500

Applicazioni del 6502, di R. Zaks - Codice 504B - L. 13.500

Impariamo a programmare in Basic con il PET/CBM, di R. Bonelli - Codice 506A - L. 10.000

Impariamo a programmare il Basic con il VIC/CBM - Codice 507A - L. 11.000

Il Timer 555, di H. Berlin - Codice 601B - L. 8.600

La progettazione dei circuiti amplificatori operazionali con esperimenti, di H. Berlin - Codice 602B - L. 15.000

La progettazione dei filtri attivi con esperimenti, di H. Berlin - Codice 603B - L. 15.000

La progettazione dei circuiti PLL con esperimenti, di H. Berlin - Codice 604H - L. 14.000

Guida ai CMOS con esperimenti, di H. Berlin - Codice 605B - L. 15.000

I tiristori - 110 progetti pratici, di R.M. Marston - Codice 606D - L. 8.000

Guida mondiale dei transistori, di TB Towers - Codice 607H - L. 20.000

Guida mondiale degli amplificatori operazionali, di T.B. Towers - Codice 608H - L. 15.000

Guida mondiale dei transistori ad effetto di campo JFET e MOS, di TB Towers - Codice 609H - L. 10.000

Amplificatori di Norton, di G. Marano - Codice 610B - L. 22.000

Manuale pratico del riparatore radio-TV, di A. Gozzi - Codice 701P - L. 18.500

Audio e Hi-Fi, di C. Brown - Codice 703D - L. 6.000

Programmazione del 6502, di Rodnay Zaks - Codice 503B - L. 22.000

Programmazione dello Z80, di Rodnay Zaks - Codice 328D - L. 24.000

Come programmare, di Jean-Claude Barbonce - Codice 511A - L. 12.000

Giochi con il 6502 - Tecniche di programmazione avanzate, di Rodnay Zaks - Codice 505B - L. 19.500

Guida al Sinclair ZX81 ZX80 e nuova ROM, di Rita Bonelli - Codice 318B - L. 16.500

DAI - Manuale del microcomputer, di R. Bonelli e C. Fiorentini - Codice 318D - L. 9.000

Programmare in Assembler, di Alain Pinaud - Codice 329A - L. 10.000

CP/M con MP/M, di Rodnay Zaks - Codice 510P - L. 22.000

Guida alla Programmazione in Assembler sul Pico Computer, di Dante Del Carso - Codice 330D - L. 9.000

Programmi Pratici in Basic, di Lon Poole - Codice 550D - L. 12.500

Soluzione di problemi con Pascal, di Benneth L. Bowles - Codice 512P - L. 28.000

Programmare in Basic, di Michel Plauin - Codice 513A - L. 8.000

Alla scoperta del TI 99/4A, di Texas Instruments - Codice 319D - L. 16.000

Manuale degli SCR TRIAC e altri tiristori, di General Electric - Codice 612P - L. 24.000

Programmare in Pascal, di Daniel-Jean David e Jean-Luc Deschamps - Codice 514A - L. 14.000

Dizionario di Informatica Italiano/Inglese/Tedesco, di Otto Vollnhals - Codice 100H - L. 45.000

76 Programmi in Basic per il vostro computer, di L. Poole M. Barchers C. Donahue - Codice 551D - L. 12.500

Potete acquistare i suddetti libri nelle migliori librerie oppure scrivendo direttamente a: **Gruppo Editoriale Jackson - Divisione Libri - Via Rosellini, 12 - 20124 Milano**

Tagliando ordine **libri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

□ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. ....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

Tagliando ordine **libri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

□ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

□ Allego assegno n° .... di L. ....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

Non si può imparare la programmazione in BASIC senza la pratica. Ecco dunque in questo volume una raccolta completa e progressiva di esercizi riguardanti matematica, gestione, ricerca operativa, gioco e statistica. Ciascun esercizio proposto comporta l'enunciazione e l'analisi del problema, la risoluzione mediante flow-chart e commenti, così come un programma che implementa la soluzione illustrato da semplici esempi rappresentativi. Questo metodo mette in grado il lettore di verificare passo passo le sue conoscenze e il livello di apprendimento raggiunto, nonchè di risolvere in modo autonomo i problemi secondo un procedimento "inverso". Cioè, come è possibile scomporre il problema in sottoproblemi ben identificati e risolvibili separatamente in modo semplice. Il lettore è così in grado di risolvere ogni difficoltà e costruire un programma modulare di facile lettura e adattabile alle specifiche applicazioni. Tutti i programmi, al fine di renderli di validità generale, sono scritti in BASIC Microsoft e girano su APPLE, PET/CBM, TRS80 e altri computer.

L. 13.000

Cod. 521A ISBN 88-70-56-126-7

70

50 ESERCIZI IN BASIC

J. P. Lamoitier

GRUPPO EDITORIALE JACKSON